



1

Semantic Framework for Mapping Object-Oriented Model to Semantic Web Languages

Petr Ježek^{1,2,*}, Roman Mouček^{1,2}

¹ Faculty of Applied Sciences, New Technologies for the Information Society, University of West Bohemia, Plzeň, Czech Republic

² Faculty of Applied Sciences, Department of Computer Science and Engineering, University of West Bohemia, Plzeň, Czech Republic

Correspondence*:

Petr Ježek

New Technologies for the Information Society, Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Univerzitní 8, 306 14 Plzeň, Czech Republic, jezekp@ntis.zcu.cz

2 ABSTRACT

3 The goal of this article is to contribute to the efficiency of research in the electrophysiology domain by improving opportunities for describing semantics of structured electrophysiological data. 4 The research efficiency can be, among other measures, expressed as an ability of data owners 5 and data users to understand experimental data in a longer term. The appropriate semantic 6 description of these data is then closely related to the successful long-term management and 7 sharing of data. Improvement of data management and sharing requires compliance with the 8 minimum standards for data formats and metadata structures. The abstract level of semantics 9 used for describing such data formats and metadata structures, which is widely discussed in the 10 community, has to be accompanied by selecting suitable semantic expressive means.

12 The article deals with and discusses two main approaches in building semantic structures for 13 metadata. It is the use of conventional data structures, repositories, and programming languages 14 on one hand and the use of formal representations of ontologies, known from knowledge representation, 15 such as description logics or semantic web languages on the other hand. Although 16 knowledge engineering offers languages supporting richer semantic means of expression and 17 technological advanced approaches, conventional data structures and repositories are still popular 18 among developers, administrators and users because of their simplicity, overall intelligibility, 19 and lower demands on technical equipment. The choice of conventional data resources and 20 repositories, however, raises the question of how and where to add semantics that cannot be 21 naturally expressed using them. As one of the possible solutions, this semantics can be added 22 into the structures of the programming language that accesses and processes the underlying 23 data.

24 To support this idea we introduced a software prototype that enables its users to add semantically 25 richer expressions into a Java object-oriented code. This approach does not burden some 26 users with additional demands on programming environment since reflective Java annotations 27 were used as an entry for these expressions. Moreover, additional semantics need not to be 28 written by the programmer directly to the code, but it can be collected from non-programmers 29 using a graphic user interface. The mapping that allows the transformation of the semantically

30 enriched Java code into the Semantic Web language OWL was proposed and implemented
31 in a library named the Semantic Framework. This approach was validated by the integration
32 of the Semantic Framework in the EEG/ERP Portal and by the subsequent registration of the
33 EEG/ERP Portal in the Neuroscience Information Framework.

34 **Keywords:** EEG/ERP Portal, electrophysiology, object-oriented code, ontology, Semantic Framework, Semantic Web

1 INTRODUCTION

35 Our research group specializes in research of brain electrical activity and operates a laboratory in which
36 the techniques and methods of electroencephalography (EEG) and event related potentials (ERP) are
37 widely used. In addition to the experimental work, which is mostly focused on recording and analysis of
38 cognitive event related potentials, we are working on the development of the software and hardware infras-
39 tructure for research in electrophysiology (**Moucek et al.**, 2014). Our experimental work is typically very
40 time consuming and vast amounts of experimental data are produced at various stages of processing from
41 data recording to their final interpretation. Based on this experience, we know that the means supporting
42 the semantic description of electrophysiological data and the software systems improving the storage,
43 management and sharing of these data (at least within a research group) contribute to long-term under-
44 standing of these data and significantly increase research efficiency. This became even more important
45 when we decided to share our data, processing steps and workflows within a wider community.

46 The subjects of long-term preservation of data, quality and range of their semantic description, and data
47 sharing itself are broadly discussed in the community. A variety of experimental approaches, techniques
48 and methods, targeted subjects, hardware and software infrastructures, etc. used during electrophysiolog-
49 ical experiments lead to the accumulation of heterogeneous data in the domain. The semantics of these
50 data means that they are accompanied by metadata that specify their meaning. Metadata can be seen as the
51 links to a specific dictionary, in which the described data are explained and defined, for example, by using
52 a plain text, sets of values or even formal logic. The richness and accuracy of the semantic content of
53 these dictionaries determines the level of semantic description of data. Then the absence of well-defined
54 metadata structures in addition to the absence of standardized and generally used data formats for raw
55 data, are the most pressing difficulties in the domain (as described e.g. in (**Teeters et al.**, 2008)).

56 In the real world electrophysiological data are stored in numerous, often proprietary, data formats and
57 annotated by metadata differently in both range and quality sense. Moreover, a number of sophisticated
58 implementations of data repositories including file systems and databases of different types are available.
59 Then the domain data are also annotated respecting limitations of the used implementations. If we take
60 into account these conceptual and technological heterogeneities and not give up efforts to increase research
61 efficiency in the field by sharing various computational resources such as raw data, metadata, processing
62 methods and workflows among laboratories, then to use a generally readable data format, create a suitable
63 semantic description of data, and develop a tool for the long-term management and sharing of data and
64 metadata in the domain are the first important steps to achieve this complex goal.

65 However, a higher level of data sharing requires compliance with the minimum standards for domain
66 data and metadata and their structures. The content of these standards and an abstract level of such de-
67 scriptions are currently broadly discussed in the community (e.g. within INCF (**INCF**, 2014) data sharing
68 activities) because a well-specified content and appropriately chosen level of abstraction could bring real
69 sharing opportunities to the domain. On the other hand, one must be careful not to discourage commu-
70 nity researchers to provide data and metadata in some standardized formats and structures by introducing
71 them too abstract semantic descriptions. Although this topic and standardization efforts themselves are
72 very important for the future of data sharing in the domain, their more extensive discussion is out of scope
73 of this article. However, it is important to introduce them since they are closely related to a more specific
74 task (Section 2.3), the description of which forms the core of this article.

75 Any proposed abstract level of the semantic description of domain metadata and their relationships has
76 to be accompanied by selecting expressive means and subsequently by selecting technologies that promote
77 data sharing. Currently there are two main approaches for building semantic structures for metadata. The
78 first approach is to use conventional data models and structures and conventional programming languages
79 providing access to data. The second approach assumes the use of formal representations of ontologies,
80 known from the knowledge representation field, such as description logic or Semantic Web languages and
81 (eventually) reasoning provided by software agents. Discussions of these approaches and their practical
82 applications have been kept for decades. They related mainly to expert systems in the past; currently
83 difficulties and perspectives of the Semantic Web languages and technologies are mostly discussed.

84 The initial concept of the Semantic Web, as was introduced in (**Berners-Lee et al.**, 2001), was based
85 on artificial intelligence techniques; knowledge bases accessed via web interfaces served for automatic
86 reasoning provided by software agents. However, this approach has been continuously changing since the
87 years 2004–2005 and the Semantic Web started to be more viewed as a large distributed database. This
88 perspective has two following fundamental aspects. The first one is known as the knowledge acquisition
89 bottleneck; acquisition of knowledge is a time consuming process that is made usually by domain experts.
90 As a result, knowledge bases are rather small comparing to conventional data repositories. This bottleneck
91 is currently even more visible in the context of big data collected from the increasing number and due to
92 technical capabilities of hardware acquisition devices. The second aspect is related to the processing of
93 conventional data that in most cases does not mean producing entirely new data by automated reasoning.
94 Data are supposed to be just queried and linked to other data. This view of data is one of the key concepts
95 of linked data (**Berners-Lee, T.**, 2006).

96 The Semantic Web expresses data by a triple-oriented language, Resource Description Framework
97 (RDF) (**Manola and Miller**, 2004). When communities working in knowledge representation and web
98 engineering started to interact more, there was a question if XML as a hierarchically oriented language or
99 RDF as a language supporting graph structures is more suitable for the representation of meaning. Finally,
100 XML is used as a mean for the serialization of RDF graphs. RDF/XML syntax as the first standardized
101 RDF syntax is still widely used. Because expressivity of RDF is limited, W3C¹ defined a more pow-
102 erful language with more capabilities for expressing meaning, Web Ontology Language (OWL) (**Dean**
103 and **Schreiber**, 2004). While RDF was accepted by a larger community, the OWL language was a large
104 burden for the practical application of the Semantic Web. This language, based on description logic, is
105 usually unintelligible to non-experts. Moreover, not many OWL constructions have been really used in
106 real applications yet. To cope with these difficulties, specific OWL dialects supporting different aspect
107 of the resulting semantic model, are currently available. The aims of the subsequent OWL2 specification
108 are to improve datatype expressivity, provide better organization of imports, and remove difficulties with
109 different versions of OWL syntaxes.

110 The concept of linked data is currently one of four ways to expose RDF data on the web. Another
111 way is based on providing SPARQL endpoints to explore data using SPARQL queries (an example is
112 the RDF platform at (**The European Bioinformatics Institute**, 2014)). SPARQL (**Prud'hommeaux and**
113 **Seaborne**, 2008) supported by W3C is the most spread standardized language for queering RDF graphs.
114 Other alternatives include publishing data directly to the web as dump files by using one of the serializa-
115 tion formats for RDF graphs and using RDFa (**W3C**, 2013) for expressing structured data in a markup
116 language (e.g., HTML). The first two approaches (linked data, SPARQL endpoints) are technologically
117 advanced, give users more opportunities to work with data, but they require specific knowledge of devel-
118 opers and administrators and place higher requirements on hardware equipment. There is also a difficulty
119 with overall availability of SPARQL endpoints; the statistics is provided by Open Knowledge Founda-
120 tion (**Open Knowledge Foundation**, 2014)). As a result, these advanced models serve mainly for initial
121 exploring the contents of the data, while in normal use the data are downloaded (as dump files) and
122 processed locally.

¹ <http://www.w3.org/>

123 Domain ontologies have been playing a significant role in information systems for a long
124 time (**Chandrasekaran et al.**, 1999) and they are well-designed for heterogeneous data description. In
125 the past ontologies were mainly created independently, they did not cope with vast amounts of data and
126 focused on logical reasoning. This approach has been changing together with the changing view of the
127 Semantic Web. Currently, ontologies are created using a bottom-up strategy to take advantage of already
128 existing data. These ontologies are then covered by upper-level ontologies or ontological background
129 models. Since for a newly created dataset we hardly find a comprehensive ontology, there are two basic
130 options (which can be used in parallel), to create an ontology for it. In the first approach, it is supposed
131 to find relevant types of objects and relations in already existing ontologies, compare them, find the most
132 suitable types and reuse them. In the second approach, new types are defined, collected and organized in
133 a newly designed ontology. The mixed strategy, when the common types are reused from other ontologies
134 and specific types are newly defined, is for example used during the development of the Ontology for
135 Experimental Neurophysiology (OEN) (**Le Franc et al.**, 2014).

136 In the electrophysiology domain at least the following initiatives are worth briefly describing. Ontology
137 for Biomedical Investigations (OBI) (**Brinkman et al.**, 2010) is an ontology for biological and clinical
138 investigation description. Its terminology contains domain-specific terms and universal terms for general
139 biological and technical usage. It uses OWL as a formal language. NEMO (**Dou et al.**, 2007) is an on-
140 tontology describing EEG, averaged EEG (ERPs), and ERP data analysis results; it lacks possibilities to
141 describe experimental protocols and restrictions. It is very complex to be immediately deployed in the
142 electrophysiology domain. On the other hand, odML (**Grawe et al.**, 2011) is an open, flexible, easy-to-
143 use and unrestricted transporting format for annotation and sharing of electrophysiology data that can be
144 implemented into any recording or management tool. The odML model for metadata defines four entities
145 (Property, Section, Value, RootSection) (**Node**, 2014). EEG/ERP Portal (EEGBase) (**Ježek and Mouček**,
146 2012) uses a classic relational database and object-oriented structures to create the domain model.

147 Looking at the semantic expressivity of ontologies from the point of view of automated processing, the
148 higher the level of formalization is, the easier it is to use the ontology for sharing and reasoning, since
149 it is more machine-processable. On the other hand, it is difficult to develop such an ontology. Moreover,
150 the use of formalisms with high expressivity leads to difficulties with decidability and computational
151 complexity when reasoning. Then these formalisms have to be limited in their expressivity to ensure that
152 the resulting ontologies are usable in practice for automated reasoning. Looking at the present state, then
153 despite the increasing popularity of storing data in repositories and retrieving them by using languages
154 providing higher semantic expressivity, most data are still stored in conventional repositories such as files
155 and relational databases (a list of DBMS ranked by their current popularity is available in (**Solid IT**,
156 2014)). Taken into account big conventional data collected from a number of hardware sensors, familiarity
157 of large groups of developers, administrators and users with conventional data repositories, and simplicity
158 of publishing RDF data (transformed from conventional data) as dump files, we can hardly anticipate a
159 substantial change in the use of current types of repositories in the near future.

160 Shared conventional data have often read-only access to third party subjects. It means that only data
161 owners are entitled to add semantics to them. Moreover, conventional relational data repositories due to
162 their limitations in semantic expressivity naturally exclude to add more complex semantic information to
163 data. Nevertheless, richer semantic descriptions can be still added later. This can be done by transforming
164 a conventional repository to a semantic repository (e.g. to RDF triple stores) or by semantic enrichment of
165 the conventional programming language that provides access to the data stored in conventional reposi-
166 tories. However, the first approach requires using the SPARQL language for later data access and retrieval.
167 Then to avoid using the Semantic Web languages and technologies to the last moment and still to cope
168 with opportunities to add richer semantic descriptions to data, it is necessary to semantically enrich a
169 conventional programming language.

170 The rest of the article is organized in the following way. The section Materials and Methods contains
171 the brief description of commonly used data models and languages both in software engineering and
172 knowledge representation fields, the state of the art in the mapping between these two approaches, and the
173 core of the article - the Semantic Framework for the mapping an object-oriented model to semantic web

174 languages. The section Results provides information about the performance and experimental evaluation
175 of the Semantic Framework. The section Discussion mainly deals with the future development of the
176 Semantic Web and related methods and technologies for data sharing.

2 MATERIALS AND METHODS

2.1 DATA MODELS AND LANGUAGES

177 Analyzing the use of conventional semantic data models (**Biller and Neuhold**, 1977; **Simsion and Witt**,
178 2004), essentially two following data modeling formalisms are widely used: the entity-relation (ERA)
179 model and object-oriented (OO) model. Newer formalisms, e.g. the Enhanced-entity-relationship (EER)
180 model, only combine these two basic formalisms. The Unified Modeling Language (UML) is the most
181 used language for modelling an application structure, behavior, architecture, business processes and data
182 structures in classic software engineering. A UML model consists of three major categories (classifiers,
183 events, behaviors) of model elements, each of which may be used to make statements about different
184 kinds of individual things within the system being modeled (**Object Management Group**, 2013).

185 The models defining object types and relations in knowledge engineering are connected with the devel-
186 opment of ontologies. Within the Semantic Web languages, RDF is a standard model for data interchange,
187 RDFS is a language for representing simple RDF vocabularies on the Web, and OWL as a computational
188 logic-based language represents rich and complex knowledge about things, groups of things, and relations
189 between things (W3C, 2012). However, the Semantic Web languages and technologies have not been
190 popular for developing application programs (**Antoniou and van Harmelen**, 2004). Within the classic
191 software engineering discipline the popularity of script-based languages in neuroinformatics has been
192 continuously rising (**Garcia and Fourcaud-Trocm**, 2009), but these languages have not been considered
193 to be suitable for the development of large systems (**Scott**, 2004). Thus an object-oriented system is still
194 the first choice when we want to design and implement a large, robust and reliable software system.

195 There is a question how we can use advantages of both the models used in software engineering and
196 knowledge representation disciplines and how we can construct a mapping between them if this is pos-
197 sible. In general, Semantic Web languages associate three types of features used in the object-oriented
198 world. They describe reality on the conceptual level independent of technological restrictions, i. e. they
199 are similar to UML representations. They also constitute a database schema for the base of facts (RDF).
200 Eventually they are processed by software tools in the implemented application, i. e. they are part of the
201 implementation.

202 It very difficult to compare UML and OWL. Although both are languages for modelling and have several
203 structural similarities, they have different capabilities and different approach to semantics. They both
204 have classes, instances, inheritance, enable defining cardinality restrictions, etc. On the other hand, OWL
205 classes are viewed like labels for concepts, while UML classes are viewed like templates for instances.
206 The most substantial differences deal with the meaning of instances (individuals in OWL) and properties.
207 In UML any class is an object that can be instantiated. This process has its semantics like assigning
208 values to attributes. Moreover, instances have a run time semantics. In OWL a class is a category, no
209 instantiation process is defined. OWL individuals, identifiers for domain things, are defined independently
210 of classes. If an OWL individual meets the criteria for the class membership, then it is a member of the
211 class. It has no state, storage or runtime semantics. UML properties always belong to a class, while OWL
212 properties are stand-alone entities. OWL properties are double types; object and datatype properties.
213 The first one links an individual to an individual and the second one links individuals to data values.
214 Understanding of a class extent is also different. While UML classes work inside a program where they
215 are defined, OWL classes provide features to share classes among domains. OWL classes may be linked
216 to a list of class descriptions (*Intersection*, *Union*, *Complement*). A property restriction, a special kind
217 of class description, describes an anonymous class, namely a class of all individuals that satisfy the
218 restriction. OWL distinguishes two kinds of property restrictions: value constraints (e.g. *allValuesFrom*,

219 *someValuesFrom*) and cardinality constraints **W3C** (2004). OWL can discipline names using *AllDifferent*,
220 *SameAs* or *DifferentFrom* constructs.

221 ODM (Ontology Definition Metamodel) (**Object Management Group**, 2009) describes the relation-
222 ships between the relevant features of UML and OWL in detail. Described differences are summarized in
223 Table 1 and shown together with a relevant Java code in Figure 1.

Table 1. OWL and UML Features Comparison

UML	OWL	Java	Comment
Class, atomic type, property OwnedAttribute	owl:Class		
instance	individual	class instance	OWL owl:individual class independent
owned attribute, association	owl:DataTypeProperty, owl:ObjectProperty	class attributes: primitive data types/objects	OWL has only global attributes
subclass, generalization	owl:subClass, owl:subProperty	<i>extends</i> , inherited classes and properties	Java does not support multiple inheritance
enumeration	owl:oneOf	<i>enum</i>	
disjoint	owl:disjointWith, owl:unionOf	One object always an instance of exactly one class, but we should pay attention to class inheritance	
multiplicity	owl:MinCardinality, owl:MaxCardinality	—	
package	ontology	<i>package</i>	
dependency	RDF:property	methods parameters or return value	
—	—	owl:intersectionOf, owl:unionOf, owl:complementOf, owl:DifferentFrom, owl:AllDifferentFrom, owl:allValuesFrom, owl:someValuesFrom, owl:SameAs	

2.2 RELATED WORK

224 Focusing on conventional data resources and programming tools, specifically a relational database and
225 an object-oriented code, we briefly describe several approaches and tools that map a relational schema or
226 an object-oriented code to the Semantic Web languages. Some of these approaches exist only as initial
227 proposals or prototypes published in scientific papers, while some of them have been really implemented
228 as available frameworks.

229 An RDF triple might be represented as a row in a table of relational database. This table has two columns,
230 corresponding to the subject and the object of the RDF triple. The name of the table corresponds to the
231 predicate of the RDF triple. The D2RQ (**Bizer and Seaborne**, 2004) platform is a system for accessing
232 relational databases as virtual, read-only RDF graph. It uses a declarative language to describe a mapping
233 between a relational database schema and RDF; the content of relational database is not replicated into an
234 RDF store. The platform **D2RQ** (2012) provides, for example, possibilities to query a non-RDF database
235 using the SparQL (**Prud'hommeaux and Seaborne**, 2008) query language, to create custom dumps of
236 the database in RDF formats, and to access information in a non-RDF database using the Apache Jena
237 API. METAMorphoses (**Švihla**, 2007) is a data transformation processor from a relational database into
238 RDF. An XML template document defines a set of mapping rules and queries for obtaining data stored in
239 a relational database.

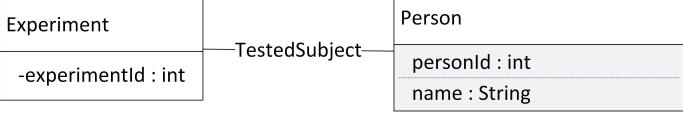
240 There are approaches and tools that provide limited possibilities to map common syntaxes of an object-
241 oriented code to an OWL representation. These tools map fundamental OWL features, it means that only
242 the basic semantic expressivity of OWL is used.

243 A mapping of OWL classes to Java Interfaces is described in (**Kalyanpur et al.**, 2004). The mapping
244 to a Java Interface instead of a common Java class enables the expression of multiple inheritance of
245 OWL properties. The back transformation is described in (**Koide et al.**, 2005), where the OWL processor
246 SWCLOS3, which is at the top of the Common Lisp Object System (CLOS), is described. Whereas CLOS
247 allows lisp programmers to develop object-oriented systems, SWCLOSS allows programmers to construct
248 domain and task ontologies in software application fields. Java2OWL-S (**Ohlbach**, 2012) is a tool which
249 is able to generate OWL directly from JavaBeans. It uses two transformations. The first transformation
250 is from JavaBeans into WSDL (Web Service Description Language). The input of this transformation is
251 formed by a Java class and the output is a temporary WSDL file. The second transformation generates
252 OWL from the WSDL file. Concerning one-side transformations (from conventional languages to Semantic
253 Web languages), these tools with common semantic expressivity work quite satisfactorily because the
254 semantic expressivity of the object-oriented code is lower than the semantic expressivity of the OWL language.
255 However, in these tools, no possibility to enrich the object-oriented code with additional semantic
256 constructs exists.

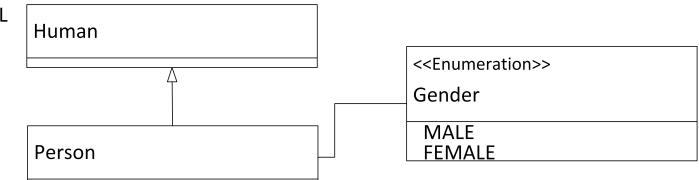
257 The Semantic Object Framework (SOF) (**Po-Huan et al.**, 2009) utilizes embedded comments in source
258 codes to describe semantic relationships between classes and attributes. The eClass (**Liu et al.**, 2007) is
259 a solution that changes Java syntax to embed semantic descriptions into an object-oriented source code.
260 These frameworks thus enrich the input object-oriented code with additional semantics using but their use
261 is difficult because they require a modified compiler and Java interpreter.

2.3 SEMANTIC FRAMEWORK

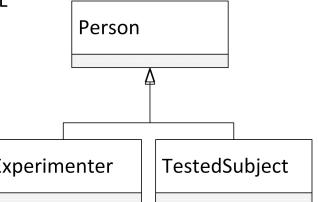
262 Drawbacks and limits of the tested frameworks motivated us to introduce a software prototype that allows
263 its users to add additional semantics directly into the Java object-oriented code. The Java code works
264 with conventional data repositories. A mapping that allows transformation of this code into the Semantic
265 Web language OWL was proposed and implemented as a library, the Semantic Framework. This solution
266 is usable by software engineers and not only by experts in the Semantic Web. It serves a community of
267 developers/researchers that develop/use object-oriented systems and need to provide an output in OWL.
268 This approach also does not burden some users with additional demands on programming environment

Java <pre>public class Experiment { private Person testedSubject; private int experimentId; }</pre>	OWL <pre><owl:Class rdf:ID="Experiment"> <owl:ObjectProperty rdf:ID="testedSubject"> <rdfs:range rdf:resource="Person"/> <rdfs:domain rdf:resource="Experiment"/> </owl:ObjectProperty> <owl:DatatypeProperty rdf:ID="experimentId"> <rdfs:range rdf:resource="integer"/> <rdfs:domain rdf:resource="Experiment"/> </owl:DatatypeProperty></pre>
UML  <pre>classDiagram Experiment "1" -- "2" Person : TestedSubject Person "1" "2" "3" personId : int Person "1" "2" "3" name : String</pre>	

(a) Definitions of classes with primitive and object properties.

Java <pre>public class Person extends Human { private Gender gender; } Public enum Gender { MALE, FEMALE }</pre>	OWL <pre><owl:Class rdf:ID="Person"> <rdfs:subClassOf rdf:resource="#Human" /> <owl:equivalentClass> <owl:Class> <owl:oneOf rdf:parseType="Collection"> <Opera rdf:about="#MALE"/> <Opera rdf:about="#FEMALE"/> </owl:oneOf> </owl:Class> </owl:equivalentClass> </owl:Class></pre>
UML  <pre>classDiagram Human "1" "2" Person Person "1" "2" "3" Gender Gender "1" "2" "3" MALE Gender "1" "2" "3" FEMALE</pre>	

(b) Definitions of inheritance and enumeration.

UML  <pre>classDiagram Person "1" "2" Experimenter Person "1" "2" TestedSubject</pre>	OWL <pre><owl:Class rdf:ID="Person"> <owl:unionOf rdf:parseType="Collection"> <owl:Class rdf:about="#Experimenter"/> <owl:Class rdf:about="#TestedSubject"/> </owl:unionOf> </owl:Class></pre>
--	--

(c) Definitions of disjoint classes.

Figure 1: Practical examples of Java code, OWL and UML Features.

269 since we use reflective Java annotations (metadata added to the Java source code and retrieved at run-
 270 time) as standard syntactic structures in Java. Moreover, additional semantics even need not to be written
 271 by the programmer directly to the code, but it can be collected from non-programmers using a graphic
 272 user interface. The presented approach is further discussed from performance (Section 3.1) and usability

273 (Section 3.2) perspectives. It was validated by the integration of the Semantic Framework in the EEG/ERP
 274 Portal, together with its registration in the Neuroscience Information Framework (NIF).

2.4 JAVABEAN TO OWL MAPPING

275 Java stores data in JavaBeans², often called Plain Old Java Objects (POJOs). The transformation of
 276 JavaBean representation into an OWL ontology is described in Definition 1.

277 DEFINITION 1. (*Extraction process from a JavaBean structure*)

278 The process is the transformation of a set of JavaBeans J to an ontology O that satisfies:

279 $\forall J_i \exists \text{OWL class } OC_i \in O$.

280 $\forall J_j$ is a superclass of $J_i \exists \text{OWL class } OC_j$ is a superclass of OWL class $OC_i \in O$.

281 $\forall \text{field} \in J f_i \exists \text{class } OC_i \in O$; its extent is a *DataType* property $\in O \Leftrightarrow J f_i$ extent is an atomic type.

282 $\forall \text{field} \in J f_i \exists \text{class } OC_i \in O$; its extent is an *Object* property $\Leftrightarrow J f_i$ extent is a class $\in J_i$.

283 $\forall \text{instance of } J_i \wedge \forall \text{field } J f_{ij} \exists \text{OWL literal } OL_{ij} \in O \hat{=} \text{field} \in J f_{ij}$.

284 An example of a mapping of a JavaBean to an OWL construct is shown in Listing 1. The JavaBean
 285 *Person* has two attributes, *researchGroups* and *firstname*. The first one is an association relation to the
 286 *ResearchGroup* class, while the second one is an atomic type. Get/set methods are omitted to keep
 287 readability. Listing 2 shows a fundamental serialization of this class into an OWL structure. The at-
 288 tribute *firstname* is serialized to a *DataTypeProperty* while the attribute *researchGroups* is serialized to an
 289 *ObjectProperty*.

Listing 1: JavaBean Person

```
290 package cz.zcu.kiv;
291
292 public class Person {
293     @Id
294     private int id;
295     private String firstname;
296     private List<ResearchGroup> researchGroups;
297 }
```

Listing 2: OWL Individual Instance

```
298 <owl:Class rdf:ID="Person">
299     <semantic:javaclass>cz.zcu.kiv.Person</semantic:javaclass>
300 </owl:Class>
301
302 <owl:Class rdf:ID="ResearchGroup">
303     <semantic:javaclass>cz.zcu.kiv.ResearchGroup</semantic:javaclass>
304 </owl:Class>
305
306 <owl:ObjectProperty rdf:ID="researchGroups">
307     <rdfs:domain rdf:resource="#this;Person"/>
308     <rdfs:range rdf:resource="#this;ResearchGroup"/>
309 </owl:ObjectProperty>
```

² JavaBeans, as reusable components, are named Java classes with class attributes which are accessed only by get/set methods.

```

311
312 <owl:DatatypeProperty rdf:id="firstname">
313   <rdfs:domain rdf:resource="#this;Person"/>
314   <rdfs:range rdf:resource="#xsd:string"/>
315 </owl:DatatypeProperty>
316
317 <owl:DatatypeProperty rdf:id="id">
318   <rdfs:domain rdf:resource="#this;Person"/>
319   <rdfs:range rdf:resource="#xsd:integer"/>
320 </owl:DatatypeProperty>
```

321 Although the described mapping works quite satisfactorily, OWL concepts described in Section 2.1 are
 322 not covered. When we want to use more capabilities of OWL, we have to enrich the object-oriented
 323 code with additional semantic expressions. Looking for a suitable way to extend a current object-
 324 oriented code, we decided to pursue a preliminary idea (**Ježek and Mouček**, 2011b) based on using
 325 Java Annotations (**MicroSystems**, 2008).

326 Java Annotations have several benefits. Firstly, they can be added, as a special form of syntactic meta-
 327 data, to a Java source code. Secondly, they are reflective, i. e. they can be embedded within the compiled
 328 code and retrieved at runtime. Moreover, Java Annotations are a part of the Standard Java Development
 329 Kit; they can be processed immediately using Java 5.0 or higher. Finally, Java Annotations are used in
 330 current software development (by several common frameworks, e.g. Spring, Hibernate, Java Persistent
 331 API); hence, software developers can work with this extension without difficulties.

332 The theoretical extraction of JavaBeans annotations and their transformations to OWL documents is
 333 formally described in Definition 2.

334 **DEFINITION 2. (Java annotation extraction process)**

335 The process is the transformation of a set of Java annotations JA to a resources R in the ontology O that
 336 satisfies:

337 $\forall JA_i \exists \text{OWL class } R_i \in O \Rightarrow JA_i \in \text{a class annotation.}$

338 $\forall JA_i \exists \text{OWL property } R_i \in O \Rightarrow JA_i \in \text{a property annotation.}$

339 An example of using annotations is given in Listing 3. The class *Person* has attributes *firstname* and
 340 *researchGroups* as defined in Listing 1. Moreover, the attribute *dateOfBirth* is added. The *firstname*
 341 attribute is defined with a value that is get from the value constraint *GivenNames* by using the *@SomeVal-*
 342 *uesFrom* annotation. The attribute *dateOfBirth* is defined with cardinality equal to 1 and the attribute
 343 *researchGroups* is defined with minimum cardinality equal to 1. In addition, the class *Person* is defined
 344 as equivalent to the class *TestedSubject* using the annotation *EquivalentClass*. The serialization of this
 345 JavaBean is shown in Listing 4. The class *Person* is a subclass of *owl:cardinality*, *owl:someValuesFrom*
 346 and *owl:minCardinality* OWL restrictions.

Listing 3: Annotated Java Bean Example

```

347 package cz.zcu.kiv;
348
349 @EquivalentClass("http://cz.zcu.kiv/TestedSubject")
350 public class Person {
351   @Id
352   private int id;
353
354   @SomeValuesFrom(stringValues = "http://cz.zcu.kiv/GivenNames")
355   private String firstname;
```

```

356     @Cardinality(1)
357     private Date dateofBirth;
358
359     @MinCardinality(1)
360     private List<ResearchGroup> researchGroups;
361 }

```

Listing 4: Annotated Java Bean OWL Serialization

```

363 <owl:Class rdf:id="Person">
364   <owl:equivalentClass>
365     <owl:Class rdf:about="http://cz.zcu.kiv/TestedSubject"/>
366   </owl:equivalentClass>
367
368   <rdfs:subClassOf>
369     <owl:Restriction>
370       <owl:cardinality rdf:datatype="xsd:int">1</owl:cardinality>
371       <owl:onProperty>
372         <owl:DatatypeProperty rdf:id="dateofBirth"/>
373       </owl:onProperty>
374     </owl:Restriction>
375   </rdfs:subClassOf>
376
377   <rdfs:subClassOf>
378     <owl:Restriction>
379       <owl:someValuesFrom>
380         <owl:DataRange>
381           <owl:oneOf rdf:parseType="Resource">
382             <rdf:rest rdf:resource="#nil"/>
383             <rdf:first rdf:datatype="xsd:string">
384               http://cz.zcu.kiv/GivenNames
385             </rdf:first>
386           </owl:oneOf>
387           <owl:DataRange>
388         </owl:someValuesFrom>
389         <owl:onProperty>
390           <owl:DatatypeProperty rdf:id="firstname"/>
391         </owl:onProperty>
392       </owl:Restriction>
393   </rdfs:subClassOf>
394
395   <rdfs:subClassOf>
396     <owl:Restriction>
397       <owl:minCardinality rdf:datatype="xsd:int">1</owl:minCardinality>
398       <owl:onProperty>
399         <owl:ObjectProperty rdf:id="researchGroups"/>
400       </owl:onProperty>
401     </owl:Restriction>
402   </rdfs:subClassOf>
403   <semantic:javaclass>cz.zcu.kiv.Person</semantic:javaclass>
404 </owl:Class>

```

Table 2. OWL Mapping of Java Annotations

Java Annotation	OWL construct
@EquivalentClass ("http://www.kiv.zcu.cz/Person")	<owl:equivalentClass rdf:resource= "http://www.kiv.zcu.cz/Person"/>
@EquivalentProperty ("http://www.kiv.zcu.cz/first_name")	<owl:equivalentProperty rdf:resource= "http://www.kiv.zcu.cz/first_name"/>
@Symmetric	<rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty"/>
@Inverse ("http://www.kiv.zcu.cz/givenname")	<owl:inverseOf rdf:resource= "http://www.kiv.zcu.cz/givenname"/>
@AllValuesFrom ("http://www.kiv.zcu.cz/#Persons")	<owl:allValuesFrom rdf:resource= "http://www.kiv.zcu.cz/#Persons"/>
@Transitive	<rdf:type rdf:resource= "http://www.w3.org/2002/07/owl#TransitiveProperty"/>
@AllDifferent ("http://www.kiv.zcu/Experiment")	<rdf:type rdf:resource= "http://www.kiv.zcu.cz/#AllDifferent"/>
@DifferentFrom ("http://www.kiv.zcu.cz/Experiment")	<owl:differentFrom rdf:resource= "http://www.kiv.zcu/Experiment"/>
@SameAs ("http://www.kiv.zcu.cz/Experiment")	<owl:sameAs rdf:resource= "http://www.kiv.zcu/Experiment"/>
@Cardinality(1)	<owl:cardinality rdf:datatype= "http://www.w3.org/2001/XMLSchema#int">1</owl:cardinality>
@MaxCardinality(1)	<owl:maxCardinality rdf:datatype= "http://www.w3.org/2001/XMLSchema#int">1</owl:maxCardinality>
@MinCardinality(1)	<owl:minCardinality rdf:datatype= "http://www.w3.org/2001/XMLSchema#int">1</owl:minCardinality>
@SomeValuesFrom ("http://www.kiv.zcu/Person")	<owl:someValuesFrom rdf:resource= "http://www.kiv.zcu/Person"/>

405 We chose the concepts that have a class and/or property extent (**Ježek and Moucek**, 2011a) and defined
 406 a set of annotations with their mapping to corresponding OWL constructs (Table 2). Most of the proposed
 407 annotations are parameterizable. Parameter values shown in Table 2 are examples; they can be changed
 408 according to the needs of a specific domain.

409 The described mapping was implemented as a library named the Semantic Framework³. It processes
 410 a set of JavaBeans as an input and produces an ontology document as an output. We did not implement
 411 the Semantic Framework from scratch, but extended and integrated already existing tools. The core of the
 412 system is extended JenaBean (**JenaBean Team**, 2010) that internally uses the Jena Framework (**Apache**
 413 **Jena Project Team**, 2011) to persist JavaBeans. Figure 2 shows the component diagram of the Semantic
 414 Framework. The first subcomponent is the extended JenaBean that reads and parses JavaBeans and related
 415 Java annotations. The output of the *Extended JenaBean* component is an internal JenaBean model that
 416 is transferred to the second, *Ontology Model Creator*, subcomponent. This subcomponent creates an
 417 ontology model using an *Ontology Model Factory* and *Jena API* methods. This ontology model extends
 418 access to the statements in a RDF data collection by adding support for constructs that are expected to
 419 be in an ontology. However, all of the state information is still encoded as RDF triples and stored in the
 420 RDF model. The resulting ontology model (in the form of an ontology document) is further processed by
 421 the last subcomponent, *OWL API* (**Horridge and Bechhofer**, 2011), which provides the ontology model
 422 in a required serialization format. The UML diagram describing the usage of the Semantic Framework is
 423 available in Figure 3.

³ The project repository is available at: <https://github.com/NEUROINFORMATICS-GROUP-FAV-KIV-ZCU/Semantic-Framework>

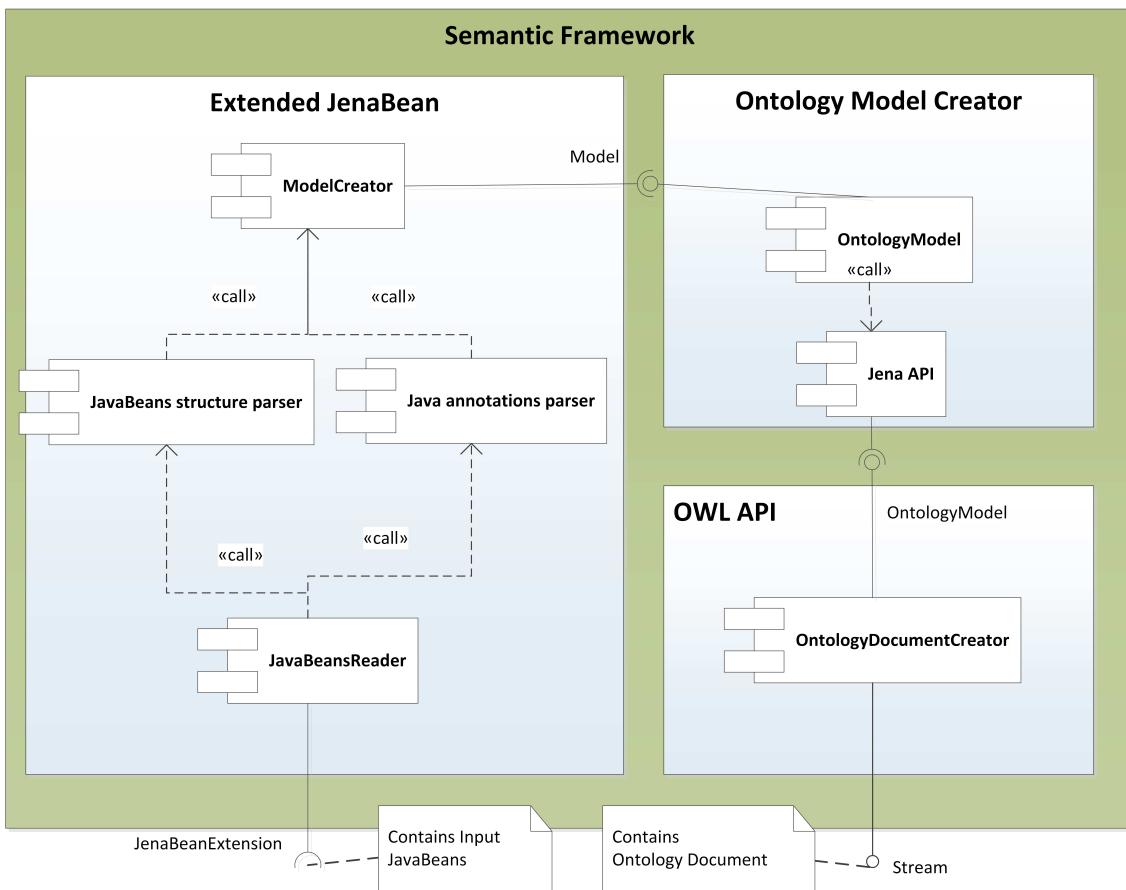


Figure 2: Component Diagram of the Semantic Framework. The Semantic Framework reads a list of input JavaBeans using an implemented reader based on Java Reflection API (SUN). This list passes through two parsers. The first one reads a JavaBean structure; the second one reads supplemented annotations. The parsers create an internal JenaBean representation. This representation is read by Jena API, which provides an ontology model processed by an OWL API. The OWL API implements existing OWL syntaxes and provides methods for serializing the ontology model.

3 RESULTS

3.1 PERFORMANCE EVALUATION

424 The time complexity of the Semantic Framework library was tested in the following way. Firstly, we
 425 prepared a set of instances of the class *Experiment*. Then, we assigned instances of the classes *Person*,
 426 *Scenario*, *Hardware*, and *Data* to each instance of the class *Experiment*. The class *Person* was extended
 427 by the set of supported annotations from Table 2. The performance tests were run ten times and the result
 428 was calculated as an average of all program runs. The time complexity of the transformational process
 429 was linear with respect to the number of instances. All the tested syntaxes are functionally equivalent;
 430 they differ only in the format of the serialized output document (Beckett, 2004).

3.2 EXPERIMENTAL EVALUATION

431 We defined a simple ontology describing the experimental work in our laboratory. Semantically, it is
 432 a modified subset of the NEMO ontology with added terms describing an experimental protocol and

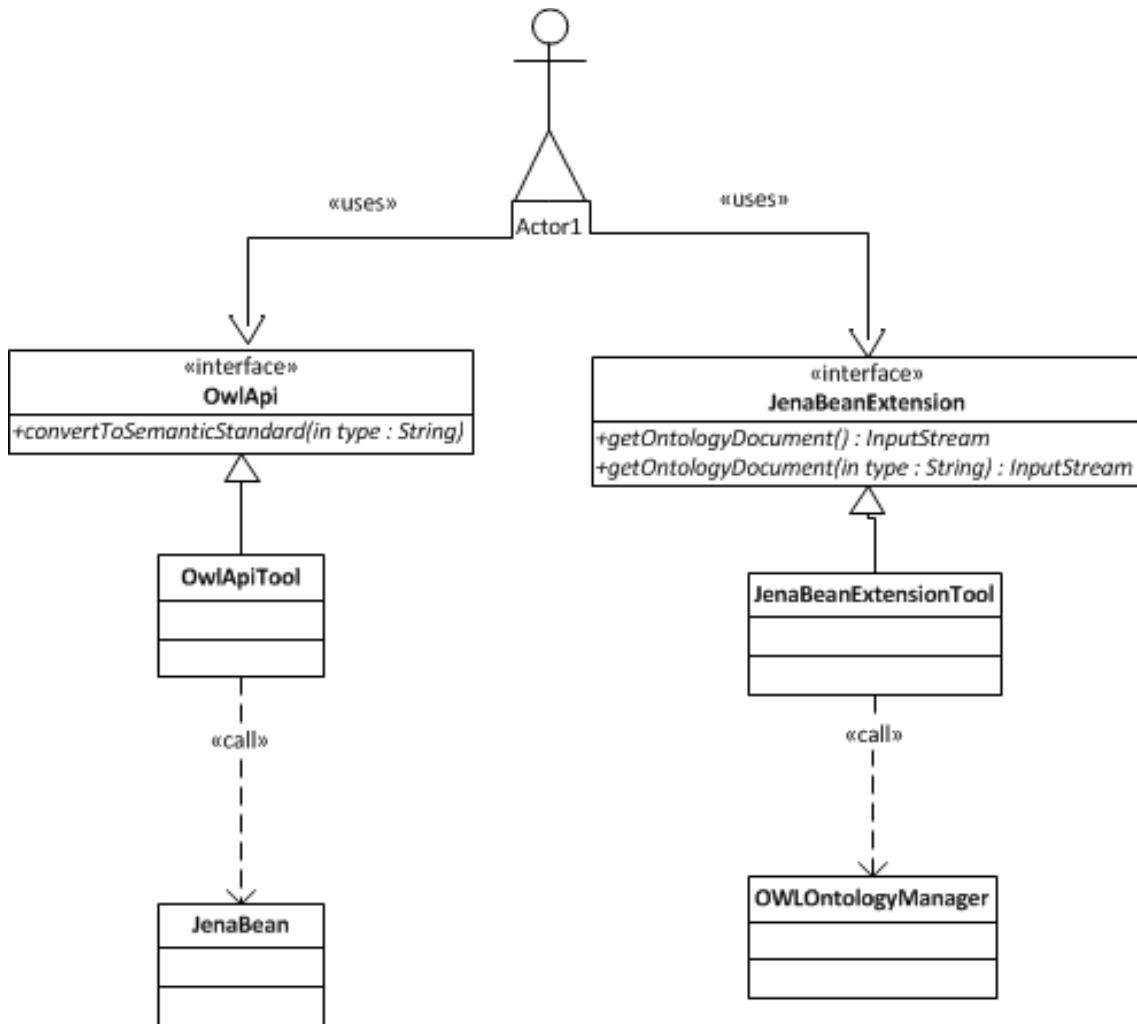


Figure 3: Class Diagram of the Semantic Framework Interface. Actor1 represents a client program. The client uses the interface *JenaBeanExtension* having the method *getOntologyDocument* that returns an ontology document in a stream. The returned stream can be serialized into supported formats by an *OwlApi* interface calling the *convertToSemanticStandard* method.

433 restrictions during an experimental session. The ontology structure corresponds to metadata collected
 434 during experiments. These metadata are divided into several semantic groups:

- 435 • Activity - describes a predefined experimental protocol. It includes information about audio and/or
 436 video stimulation, instructions given to tested subjects, detailed descriptions of stimuli (target vs.
 437 non-target, timing), etc.
- 438 • Environment - describes surrounding conditions such as weather, daytime or room temperature.
- 439 • Tested subject - includes information about the tested subject such as laterality, education, age, gender,
 440 diseases, and disability.
- 441 • Hardware equipment - describes e.g. the type, producer, and serial number of the hardware used.
- 442 • Software equipment - describes software used during the experiment. It includes e.g. the name of the
 443 software, version, manufacturer, and configuration files if they are used.

- 444 • Used electrodes - describes the type, impedance, location, used system, and fixation of the electrodes.
- 445 • Data digitalization - describes a set of parameters that influence conversion of data using a specific
- 446 analogue-digital converter. It includes filtration, sampling frequency, and band-pass.
- 447 • Signal analysis - describes basic analytic steps during the EEG/ERP signal processing. It includes the
- 448 determination of the length of the pre- and post-stimulus part of the signal, number of epochs, and
- 449 text description of the signal-processing procedure.
- 450 • Data presentation - describes experimental results or assumptions needed to reproduce an experiment.
- 451 It includes averaged ERP waves (images of averaged waves), grand averages (images of grand aver-
- 452 ages), evolution of the ERP signal in time and space (images showing the ERP signal propagation over
- 453 the scalp), waves description (description of well-known or new waves formed during the study), and
- 454 link to raw experimental data.
- 455 • Signal artifact - contains information describing a compensation method that prevents formation of
- 456 artifacts. When a method for removing artifacts is used, its description is also placed there. When
- 457 some artifact totally degrades the signal, the experimenter can define conditions when it is possible to
- 458 assume that the signal is totally useless.

459 This simple ontology was built within the development of the EEG/ERP Portal (EEGBase) (**Ježek and**
 460 **Mouček, 2012**), which is a web application (**Neuroinformatics group, University of West Bohemia,**
 461 2014) for the storage, long-term management, and sharing of electrophysiology data. The data layer of
 462 the EEG/ERP Portal is implemented using a relational database (Oracle 11g) and POJOs. An object-
 463 relational mapping (ORM) is ensured by the Hibernate framework (**Bauer and King, 2006**). The internal
 464 structure (classes and their relationships, annotations) of the data layer is implemented according to the
 465 defined ontology. The application layer was developed using the Spring Framework; the presentation layer
 466 uses Apache Wicket. Upload of data and metadata is ensured via a set of predefined web forms.

467 The Semantic Framework was integrated into the EEG/ERP Portal (Figure 4). The internal logic⁴ calls
 468 a Semantic Framework API (the UML diagram describing the Semantic module API in the EEG/ERP
 469 Portal is shown in Figure 5) using a built-in timer at regular intervals. The created ontology document is
 470 stored in a temporary file that is further serialized into a required syntax. Syntaxes *RDF/XML*, *OWL/XML*,
 471 *RDF/XML-ABBREV*, *N-TRIPLE*, *TURTLE*, *N3*, *N3-PP*, *N3-PLAIN*, and *N3-TRIPLE* are currently sup-
 472 ported. The *SemanticMultiController* is listening on a specific URL with the *GET* parameter. For instance,
 473 when a reasoner visits the URL <http://eegdatabase.kiv.zcu.cz/seman tic/getOntology.html?type=turtle>, it
 474 obtains the OWL document in the *turtle* syntax. The output ontology document is valid according to W3C
 475 specification. It can be proved by its visualization in Protégé (shown in Figure 6).

476 The generated OWL documents are typically used when registering the EEG/ERP Portal with other
 477 providers of neuroinformatics services. We successfully used the Neuroscience Informational Framework
 478 (NIF) (**Gardner et al., 2008**). The NIF framework provides a unified interface for accessing neurophys-
 479 iological data through resources described by ontology web languages (**Gupta et al., 2008**). NIF uses
 480 a proprietary framework *DISCO* (**Marenco et al., 2010**). It is an XML-based script containing a static de-
 481 scription of the registered resource. The dynamic content is accessed through a generated ontology. The
 482 structure of metadata instances is stored in an *Interoperability XML* file that is a part of the DISCO proto-
 483 col. The interoperability file is stored in the root directory of the EEG/ERP Portal together with generated
 484 DISCO files. The NIF framework reloads it at regular intervals. It enables dynamic access to the content
 485 of the EEG/ERP Portal. Figure 7 shows experiments listed through the NIF registry. Currently more than
 486 100 experiments are available in the NIF registry and new ones are being gradually added.

⁴ Spring MVC is used. It practically implements a Model-View-Controller design pattern.

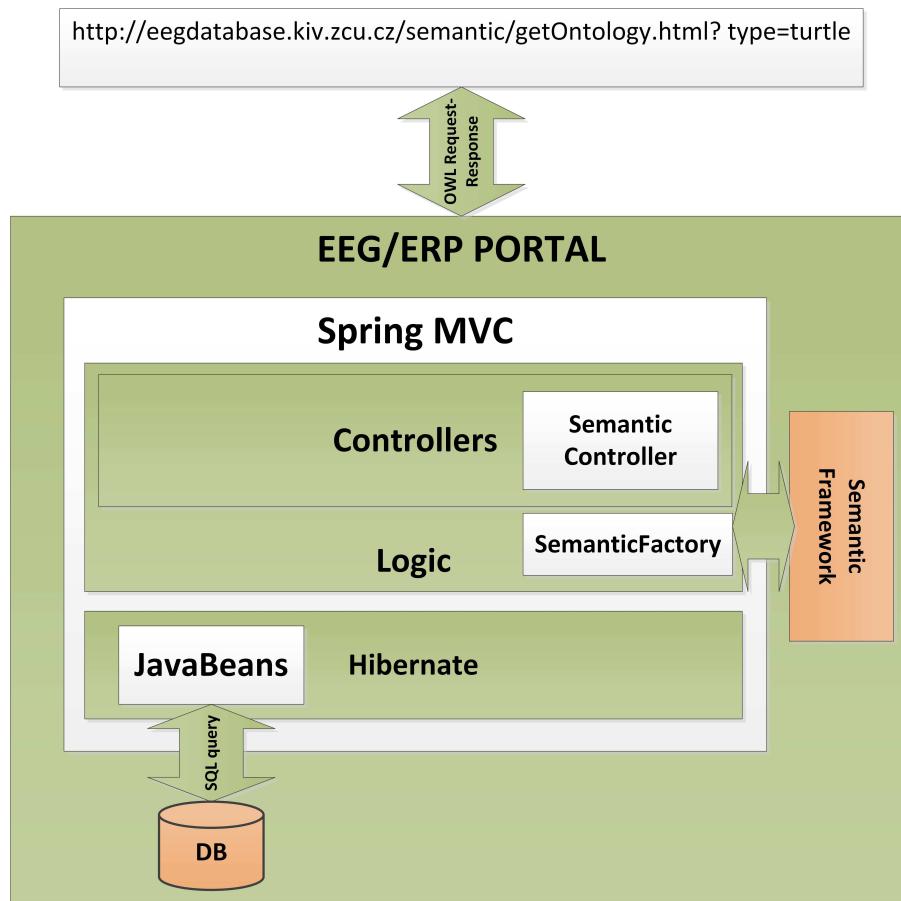


Figure 4: Integration of the Semantic Framework in the EEG/ERP Portal. The EEG/ERP Portal is a layered architecture designed according to a Model-View-Controller design pattern. The data layer obtains Javabeans from a relational database. The application layer is controlled by the set of *Controllers* which process user requests originated from a web browser. The specific controller (*Semantic Controller*) processes ontology document requests. This controller calls the integrated Semantic Framework through *Semantic Factory* and returns a serialized ontology document to the user's web browser.

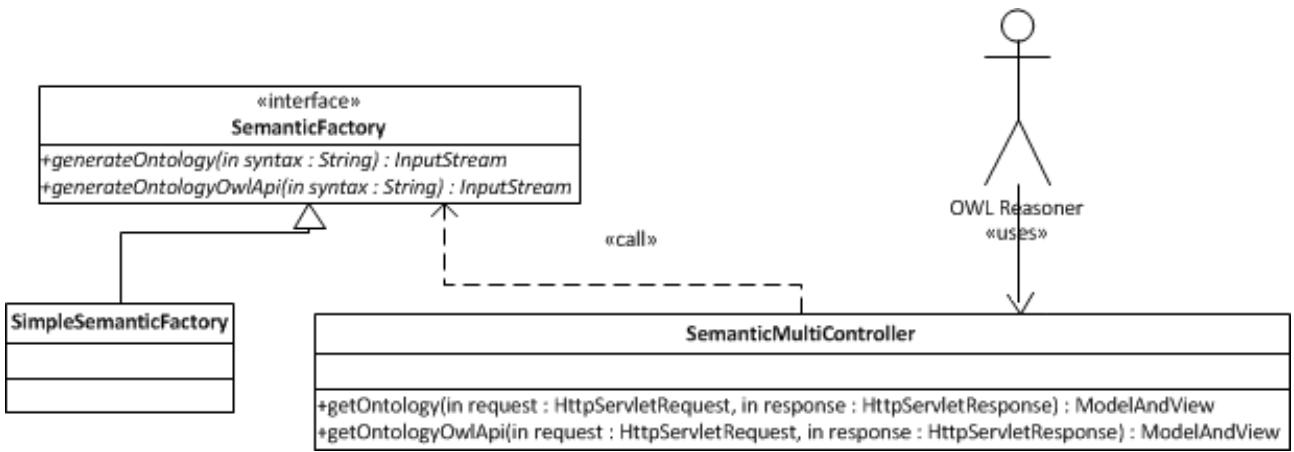


Figure 5: Semantic Framework Integration. A user (e.g. OWL reasoner) works with the `SemanticMulti-controller` interface. This controller has two methods, `getOntology` and `getOntologyOwlApi`. A required output syntax is passed to the methods as a part of the `HttpServletRequest` parameter. The Semantic Framework is called within the methods of the implemented `SemanticFactory` Interface.

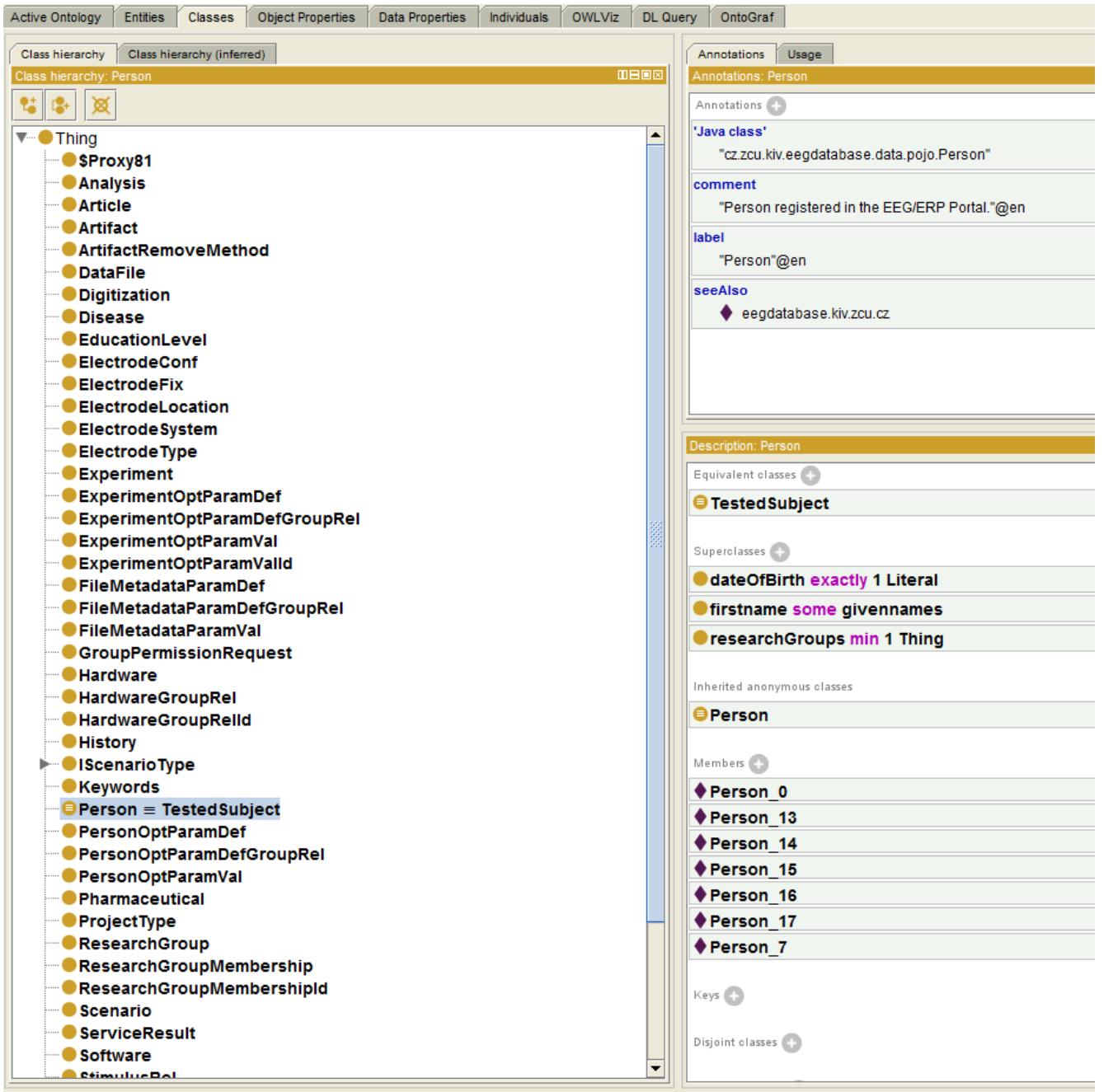


Figure 6: Protégé Visualization. The left column shows ontology classes. All classes are subclasses of a superclass *Thing*. The right window shows the class *Person*. The class descriptions (*EquivalentClass*) and properties (e.g. *researchGroups* - each person is a member of at least one research group) are transformed from the enriched JavaBean (see Listing 3). The class instances (e.g. *Person_0*) are originally obtained from the relational database.

Subject	Scenario Title	Gender	Year Of Birth	Experimental Hardware	Length Of Recording	Description
EEGbase_subject_person_1 45	Test	female	1960	VAmp	1561 minutes	ERP team test
EEGbase_subject_person_3 1	Drinkability scenario	female	1959	Blue EEG cap	45 minutes	drinkability scenario - presentation
EEGbase_subject_person_8 0	Steady State 3 Hz	female	1955	Brain Vision Amplifier,Red EEG Cap	10 minutes	Flashing LED with 3 Hz
EEGbase_subject_person_2 4	P3 LED (D,T,N)	male	1949	Blue EEG cap,Brain Vision Data Amplifier,Brain Vision Data Exchange	20 minutes	Classic oddball experiment.
EEGbase_subject_person_1 24	Driver's_Attention_Rada	female	1974	Brain Vision Amplifier,Red EEG Cap	60 minutes	Driver's attention during monotonous driving and audio stimulation (ERP experiment)
EEGbase_subject_person_1 26	Driver's_Attention_Rada	male	1981	Brain Vision Amplifier,Red EEG Cap	60 minutes	Driver's attention during monotonous driving and audio stimulation (ERP experiment)
EEGbase_subject_person_1 40	P3 LED (D,T,N)	male	1951	Blue EEG cap,Brain Vision Data Amplifier,Brain Vision Data Exchange	20 minutes	Classic oddball experiment.
EEGbase_subject_person_5 6	ERP_Gorschenek	male	1966	Red EEG Cap	40 minutes	Driver's attention - audio and visual stimulation
EEGbase_subject_person_1 28	Driver's_Attention_Rada	male	1987	Brain Vision Amplifier,Red EEG Cap	60 minutes	Driver's attention during monotonous driving and audio stimulation (ERP experiment)
EEGbase_subject_person_8 0	P3 LED (D,T,N)	female	1955	Brain Vision Amplifier,Red EEG Cap	20 minutes	Classic oddball experiment.
EEGbase_subject_person_3 6	Drinkability scenario	male	1981	Blue EEG cap	45 minutes	drinkability scenario - presentation
EEGbase_subject_person_4 8	ERP_Renicha_scenario	male	1983	Red EEG Cap	40 minutes	audio stimulation - driver's attention
EEGbase_subject_person_1 14	Driver's_Attention_Rada	male	1979	Brain Vision Amplifier,Red EEG Cap	60 minutes	Driver's attention during monotonous driving and audio stimulation (ERP experiment)
EEGbase_subject_person_1 15	P3 LED (D,T,N)	female	1988	Brain Vision Amplifier,Red EEG Cap,Brain Vision Data Exchange	20 minutes	Classic oddball experiment.
EEGbase_subject_person_1 42	P3 LED (D,T,N)	female	1973	Brain Vision Amplifier,Red EEG Cap,Brain Vision Data Exchange	20 minutes	Classic oddball experiment.
EEGbase_subject_person_8 1	Loss of Working Memory (Train)	female	1978	Red EEG Cap	10 minutes	Subject is watching at movie with 1st person view from train cabin. Three LEDs are flashing at the bottom ...[more]
EEGbase_subject_person_3 3	P3 LED (D,T,N)	male	1954	Blue EEG cap,Brain Vision Data Amplifier,Brain Vision Data Exchange	20 minutes	Classic oddball experiment.

Figure 7: EEG/ERP Portal in the NIF Registry. The list of experiments stored in the EEG/ERP Portal is shown in the NIF registry at link <https://www.neuinfo.org/mynif/search.php?q=eeegbase&t=indexable&list=cover&nif=nif-0000-08190-1>. The list contains direct hyperlinks to the experiments stored in the EEG/ERP Portal. When a user clicks on a hyperlink, he/she is redirected to the EEG/ERP Portal.

4 DISCUSSION

487 This article described the possible approaches to the semantic enrichment of structured electrophysiology
488 data. Different views of software engineering and knowledge representation communities on data
489 modelling, reasonable range of semantic descriptions, and used languages and technologies were briefly
490 introduced. The Semantic Web has become (after 13 years of its existence) a kind of connection between
491 these communities. Currently, the real benefits of the Semantic Web can be found in the concept of linked
492 data that is technologically well supported. However, in general the Semantic Web technologies are not
493 mature, they are often computationally demanding and the community of developers and administrators
494 who develop/maintain/administrate them is significantly smaller than communities interested in "conven-
495 tional" programming languages and tools. On the other hand, it is worthwhile to use and promote the
496 Semantic Web languages, standards and technologies that can bring to neuroinformatics applications the
497 opportunity to use higher semantic expressivity.

498 Based on these assumptions we developed a software prototype, the Semantic Framework library, that
499 connect conventional technologies and programming tools (relational database, domain-independent Java-
500 based systems) with the languages and technologies of the Semantic Web (RDF, OWL, JenaBean, Jena,
501 OWL API). The most important contribution is a transformational mechanism that maps common Java-
502 aBeans accessing data stored in a relational database into OWL individuals. In addition, the semantic
503 diversity that exists due to the different semantic expressivity of the object-oriented model and the Se-
504 mantic Web languages was partly addressed using a custom annotation-based approach. Java annotations
505 are enriched by additional semantic constructions that are also transformed to the resulting OWL ontology
506 document. This approach using annotations replaces the traditional modeling of ontologies in an ontolog-
507 ical language. The Semantic Framework was integrated in the EEG/ERP Portal and enabled dynamic
508 generation of the output ontology document. This document is used by the Neuroscience Information
509 Framework to provide a content of the EEG/ERP Portal through its interface.

510 Although it is difficult to predict the future development of the Semantic Web, at least it can be ex-
511 pected that new proposals for standards will appear and the software tools that will be developed become
512 more mature and stable. These predictions concerning the future development of the Semantic Web (and
513 standards and tools for semantic descriptions in general) are important for our decisions regarding the
514 development of the EEG/ERP Portal and the Semantic Framework itself.

515 One of the possible challenges is replacement of the relational database with a NoSQL database for
516 storing experimental metadata. Relational-databases are inflexible when structure modifications are re-
517 quired, while NoSQL databases provide higher scalability and availability because of their free schema.
518 NoSQL databases having key-value organization can easily store RDF triples (**Papailiou et al.**, 2012).
519 There are initiatives, e.g. (**Ebel and Hulin**, 2012), that investigate the transformation of a common re-
520 lational database to a NoSQL database. Currently we replaced a part of the relational database for the
521 NoSQL database ElasticSearch.

522 Another challenge is to integrate a standardized data format and metadata structures into the EEG/ERP
523 Portal. We participate in these standardization activities within INCF Electrophysiology Task Force and
524 within the group developing an experimental ontology for neurophysiology (**Ontology for Experimental**
525 **Neurophysiology Working Group**, 2013). Even the partial outcomes of these groups are continuously
526 integrated into the EEG/ERP Portal.

527 The presented approach was used and tested in the electrophysiology domain, but the mapping
528 mechanism implemented in the Semantic Framework can be easily applied to other domains.

5 ACKNOWLEDGEMENTS

529 This work was supported by the European Regional Development Fund (ERDF), Project "NTIS - New
530 Technologies for Information Society", European Centre of Excellence, CZ.1.05/1.1.00/02.0090.

REFERENCES

- 531 Antoniou, G. and van Harmelen, F. (2004), A Semantic Web Primer (Cooperative Information Systems
 532 series) (The MIT Press)
- 533 Apache Jena Project Team (2011), Apache Jena - A free and open source Java framework for building
 534 Semantic Web and Linked Data applications
- 535 Bauer, C. and King, G. (2006), Java Persistence with Hibernate (Manning Publications Co., Greenwich,
 536 CT, USA)
- 537 Beckett, D. (2004), RDF/xml syntax specification (revised), W3C recommendation, W3C
- 538 Berners-Lee, T., Hendler, J., and Lassila, O. (2001), The semantic web, *Scientific American*, 284, 5, 34–43
- 539 Berners-Lee, T. (2006), Designed Issues:LinkedData
- 540 Biller, H. and Neuhold, E. (1977), Architecture and models in data base management systems: Proceed-
 541 ings of the IFIP Working Conference on Modelling in Data Base Management Systems (distributors for
 542 the U.S.A. and Canada, Elsevier/North Holland)
- 543 Bizer, C. and Seaborne, A. (2004), D2RQ-treating non-RDF databases as virtual RDF graphs, in
 544 Proceedings of the 3rd International Semantic Web Conference (ISWC2004) (Citeseer)
- 545 Brinkman, R., Courtot, M., Derom, D., Fostel, J., He, Y., Lord, P., et al. (2010), Modeling biomedical
 546 experimental processes with OBI, *Journal of Biomedical Semantics*, 1, Suppl 1, S7+, doi:10.1186/
 547 2041-1480-1-s1-s7
- 548 Chandrasekaran, B., Josephson, J., and Benjamins, V. (1999), What are ontologies, and why do we need
 549 them?, *Intelligent Systems and their Applications, IEEE*, 14, 1, 20–26, doi:10.1109/5254.747902
- 550 D2RQ (2012), D2RQ
- 551 Dean, M. and Schreiber, G. (2004), OWL web ontology language reference, W3C recommendation, W3C
- 552 Dou, D., Frishkoff, G. A., Rong, J., Frank, R., Malony, A. D., and Tucker, D. M. (2007), Development of
 553 neuroelectromagnetic ontologies(nemo): a framework for mining brainwave ontologies, in P. Berkhin,
 554 R. Caruana, and X. Wu, eds., Proceedings of the 13th ACM SIGKDD International Conference on
 555 Knowledge Discovery and Data Mining, San Jose, California, USA, August 12–15, 2007 (ACM), 270–
 556 279, doi:<http://doi.acm.org/10.1145/1281192.1281224>
- 557 Ebel, M. and Hulin, M. (2012), Combining relational and semi-structured databases for an inquiry applica-
 558 tion, in G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, eds., Multidisciplinary Research and
 559 Practice for Information Systems, volume 7465 of *Lecture Notes in Computer Science* (Springer Berlin
 560 Heidelberg), 73–84, doi:10.1007/978-3-642-32498-7__6
- 561 Garcia, S. and Fourcaud-Trocm, N. (2009), Openelectrophy: an electrophysiological data- and analysis-
 562 sharing framework, *Frontiers in Neuroinformatics*, 3, 14, doi:10.3389/neuro.11.014.2009
- 563 Gardner, D., Akil, H., Ascoli, G., Bowden, D., Bug, W., Donohue, D., et al. (2008), The neuroscience
 564 information framework: A data and knowledge environment for neuroscience, *Neuroinformatics*, 6,
 565 149–160, doi:10.1007/s12021-008-9024-z
- 566 Grewe, J., Wachtler, T., and Benda, J. (2011), A bottom-up approach to data annotation in neurophysiol-
 567 ogy, *Frontiers in Neuroinformatics*, 5, 16, doi:10.3389/fninf.2011.00016
- 568 Gupta, A., Bug, W., Marenco, L., Qian, X., Condit, C., Rangarajan, A., et al. (2008), Feder-
 569 ated access to heterogeneous information resources in the neuroscience information framework (nif),
 570 *Neuroinformatics*, 6, 205–217, 10.1007/s12021-008-9033-y
- 571 Horridge, M. and Bechhofer, S. (2011), The owl api: A java api for owl ontologies, *Semantic Web*, 2, 1,
 572 11–21
- 573 INCF (2014), International neuroinformatics coordinating facility
- 574 JenaBean Team (2010), jenabean - Project Hosting on Google Code
- 575 Ježek, P. and Mouček, R. (2011a), Semantic web in eeg/erp portal: Ontology development and nif reg-
 576 istration, in Biomedical Engineering and Informatics (BMEI), 2011 4th International Conference on,
 577 volume 4, volume 4, 2058–2062, doi:10.1109/BMEI.2011.6098757
- 578 Ježek, P. and Mouček, R. (2011b), Transformation of object-oriented code into semantic web using java
 579 annotations., in R. Zhang, J. Cordeiro, X. Li, Z. Zhang, and J. Zhang, eds., ICEIS (4) (SciTePress),
 580 207–210

- 581 Jezek, P. and Moucek, R. (2012), System for EEG/ERP Data and Metadata Storage and Management,
 582 *Neural Network World*, 22, 3, 277–290
- 583 Kalyanpur, A., Pastor, D. J., Battle, S., and Padgett, J. A. (2004), Automatic Mapping of OWL Ontolo-
 584 gies into Java, in F. Maurer and G. Ruhe, eds., Proceedings of the 16th Int'l Conference on Software
 585 Engineering & Knowledge Engineering (SEKE'2004) (Banff, Alberta, Canada), 98–103
- 586 Koide, S., Aasman, J., and Haflisch, S. (2005), Owl vs. object orientated programming, in in International
 587 Semantic Web Conference, Workshop1: Semantic Web Enabled Software Engineering
- 588 Le Franc, Y., Bandrowski, A., Bruha, P., Papež, V., Grewe, J., Moucek, R., et al. (2014), Describing
 589 neurophysiology data and metadata with oen, the ontology for experimental neurophysiology, *Frontiers*
 590 in *Neuroinformatics*, , 44, doi:10.3389/conf.fninf.2014.18.00044
- 591 Liu, F., Wang, J., and Dillon, S. T. (2007), Web information representation, extraction and reasoning
 592 based on existing programming technology, in Computational Intelligence 37, 147–168
- 593 Manola, F. and Miller, E., eds. (2004), RDF Primer, W3C Recommendation (World Wide Web
 594 Consortium)
- 595 Marenco, L., Wang, R., Shepherd, G., and Miller, P. (2010), The nif disco framework: Facili-
 596 tating automated integration of neuroscience content on the web, *Neuroinformatics*, 8, 101–112,
 597 doi:10.1007/s12021-010-9068-8
- 598 MicroSystems, S. (2008), Annotations (the java tutorials, learning the java language, classes and objects)
- 599 Moucek, R., Bruha, P., Jezek, P., Mautner, P., Novotny, J., Papez, V., et al. (2014), Software and hardware
 600 infrastructure for research in electrophysiology, *Frontiers in Neuroinformatics*, 8, 20, doi:10.3389/fninf.
 601 2014.00020
- 602 Neuroinformatics group, University of West Bohemia (2014), EEG/ERP Portal (EEGBase)
- 603 Node, G. N. (2014), odml data model
- 604 Object Management Group (2009), Ontology definition metamodel (omg) version 1.0, Technical Report
 605 formal/2009-05-01, Object Management Group
- 606 Object Management Group (2013), OMG Unified Modeling Language (OMG UML), version 2.5
- 607 Ohlbach, H. J. (2012), Java2owl: A system for synchronising java and owl, in J. Filipe and J. L. G. Dietz,
 608 eds., KEOD (SciTePress), 15–24
- 609 Ontology for Experimental Neurophysiology Working Group (2013), Ontology for experimental neuro-
 610 physiology
- 611 Open Knowledge Foundation (2014), Availability of SPARQL Endpoint
- 612 Papailiou, N., Konstantinou, I., Tsoumakos, D., and Koziris, N. (2012), H2rdf: adaptive query processing
 613 on rdf data in the cloud., in Proceedings of the 21st international conference companion on World Wide
 614 Web (ACM, New York, NY, USA), WWW '12 Companion, 397–400, doi:10.1145/2187980.2188058
- 615 Po-Huan, C., Chi-Chuan, L., and Kuo-Ming, C. (2009), Integrationg semantic web and object-oriented
 616 programming for cooperative design, *Journal of University Computer Science*, vol. 15, no. 9
- 617 Prud'hommeaux, E. and Seaborne, A. (2008), Sparql query language for rdf, W3c recommendation, W3C
- 618 Scott, T. (2004), Python: the good, the bad, and the not ugly: conference workshop, *J. Comput. Sci. Coll.*,
 619 20, 1, 288–290
- 620 Simsion, G. and Witt, G. (2004), Data Modeling Essentials, Third Edition (Morgan Kaufmann)
- 621 Solid IT (2014), DB Engines - Knowledge Base of Relational and NoSQL Database Management Systems
- 622 SUN (2006), Java Tutorial Trail: The Reflection API, SUN Microsystems
- 623 Teeters, J., Harris, K., Millman, K., Olshausen, B., and Sommer, F. (2008), Data sharing for computational
 624 neuroscience, *Neuroinformatics*, 6, 1, 47–55, doi:10.1007/s12021-008-9009-y
- 625 The European Bioinformatics Institute (2014), RDF Platform
- 626 Švihla, M. (2007), Transforming Relational Data into Ontology Based RDF (CTU Prague), thesis
- 627 W3C (2004), OWL Web Ontology Language Reference
- 628 W3C (2012), Web Ontology Language
- 629 W3C (2013), RDFa Core 1.1 - Second Edition