OBIS: Ontology-Based Information System Framework

Kārlis Čerāns[1], Aiga Romāne[1]

karlis.cerans@lumii.lv, aiga.romane@inbox.lv
Institute of Mathematics and Computer Science, University of Latvia
Raina blvd. 29, Riga, LV-1459, Latvia

Abstract. We demonstrate a framework for automated information system generation from a given data ontology describing either an existing data set, or a data set to be filled up and maintained using the created information system. The IS data are stored into real or virtual RDF data stores and are accessed by means of SPARQL queries. The data ontology structure information and annotations allow automated generation of IS reflecting the data structure specified in the ontology.

Keywords: Ontologies, RDF schemas, automated information system generation, user interface annotations.

1 Introduction

The semantic web standards, including RDF [1,2], OWL [3] and SPARQL [4] define information representation and querying infrastructure that is basis for Semantic Web [5] and Linked Data [6], as well as enterprise-level semantic technology use. The semantic technologies offer much higher-level view on data than do the classic relational databases (RDB) with their corresponding SQL query language thus raising a hope of more direct involvement of various domain experts in data set definition, exploration and analysis. There are both a W3C standard R2RML [7], as well as numerous developments (e.g. Virtuoso RDF Views [8], D2RQ [9], ontop [10] and RDB2OWL [11]) for mapping relational database information into the semantic technology landscape. The RDF data stores such as Virtuoso [8] and Stardog [12] provide native RDF data storage so facilitating the RDF data availability. For the semantic information landscape implementation there is also a need of tools allowing creation, access and analysis of the available data.

We demonstrate here a system for automated information system user interface creation from the structure of data described as annotated OWL ontology thus allowing browsing the available RDF data, as well as editing the data, if the corresponding RDF data store allows for data updates and data entry. There are two basic use cases of the OBIS Framework: (i) to browse and analyze data in existing

[1] Partially supported by Latvian State Research programme NexIT project No.1 'Technologies of ontologies, semantic web and security'

real or virtual RDF databases, including the RDF databases that are conceptual representations of relational database data [13,14], as well as arbitrary SPARQL endpoints that have a corresponding data schema description available, and (ii) to manage information system data primarily stored into a RDF database (including data inserting and updating); thus creating a SPARQL-enabled conceptual data store.

We structure the presentation into two main sections – first is explained the basic OBIS application generation, followed by application tuning possibility description. The conceptual foundation of the OBIS system has been described in [15], the current paper reports on new basic functionality (e.g. related items view), as well as entirely new ontology annotation framework.

2 OBIS: Basic Working Scheme

The ontology-based information system creation in OBIS starts with obtaining or designing of data ontology reflecting the structure of the data underlying the information system. Figure 1 contains an example mini-University ontology in the OWLGrEd[2] ontology editor notation [16] describing OWL classes as UML classes, OWL object properties as role links between property domain and range classes and OWL data properties as their domain class attributes.

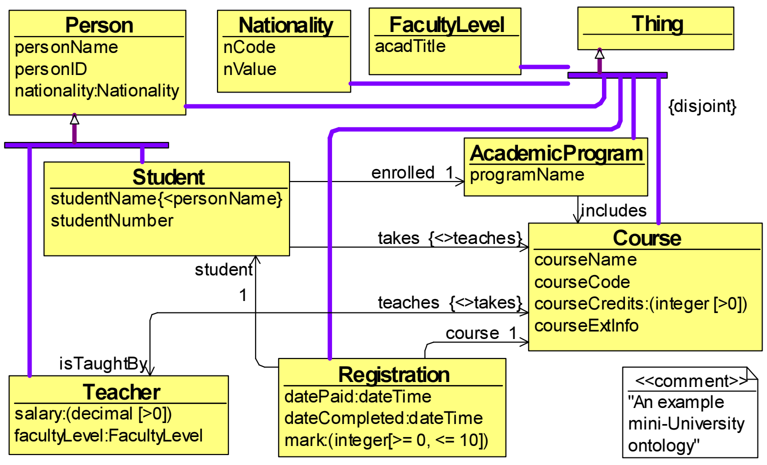


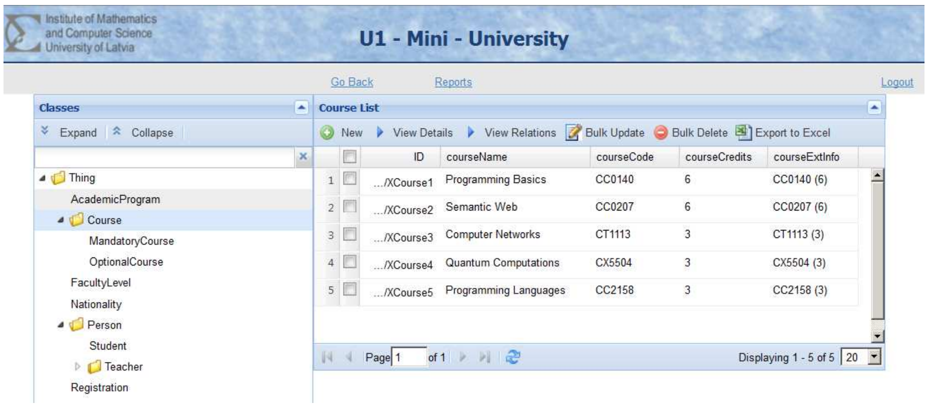
Fig. 1. Example: the OWLGrEd notation for a mini-university ontology

With the OBIS framework up and running on a web-server the user creates a new application in the OBIS Application Manager. The user then fills in the application name as well as ontology namespace (from the data ontology) and data namespace (not relevant for read-only ontology data access), chooses the repository type (e.g. Virtuoso server[3], a SPARQL-endpoint or a Jena TDB store) and connection

[2] The editor can be downloaded from http://owlgred.lumii.lv

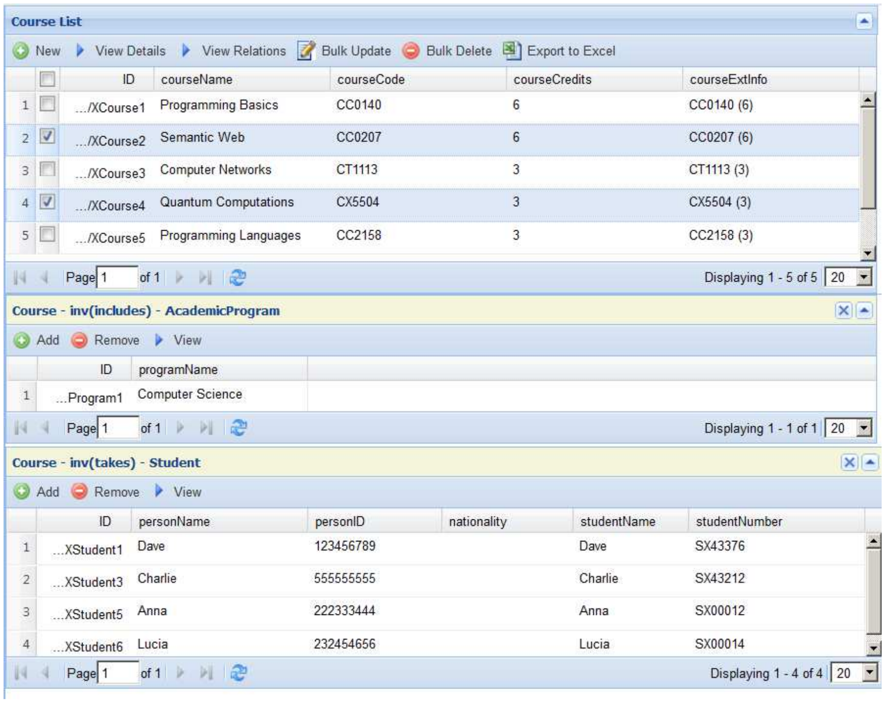
[3] To be downloaded from e.g. http://virtuoso.openlinksw.com/ and installed separately

parameters. After saving the created basic application information and loading the data ontology the concrete application based on the data ontology can be generated and run. Figure 2 shows mini-University example case of the obtained application working interface, consisting of a class tree browser with the option to select a class and obtain a table view of its instance attributes.



	ID	courseName	courseCode	courseCredits	courseExtInfo
1	.../XCourse1	Programming Basics	CC0140	6	CC0140 (6)
2	.../XCourse2	Semantic Web	CC0207	6	CC0207 (6)
3	.../XCourse3	Computer Networks	CT1113	3	CT1113 (3)
4	.../XCourse4	Quantum Computations	CX5504	3	CX5504 (3)
5	.../XCourse5	Programming Languages	CC2158	3	CC2158 (3)

Fig. 2. Example: the generated mini-University information system: the instance table view for Course class.



		ID	courseName	courseCode	courseCredits	courseExtInfo
1	☐	.../XCourse1	Programming Basics	CC0140	6	CC0140 (6)
2	☑	.../XCourse2	Semantic Web	CC0207	6	CC0207 (6)
3	☐	.../XCourse3	Computer Networks	CT1113	3	CT1113 (3)
4	☑	.../XCourse4	Quantum Computations	CX5504	3	CX5504 (3)
5	☐	.../XCourse5	Programming Languages	CC2158	3	CC2158 (3)

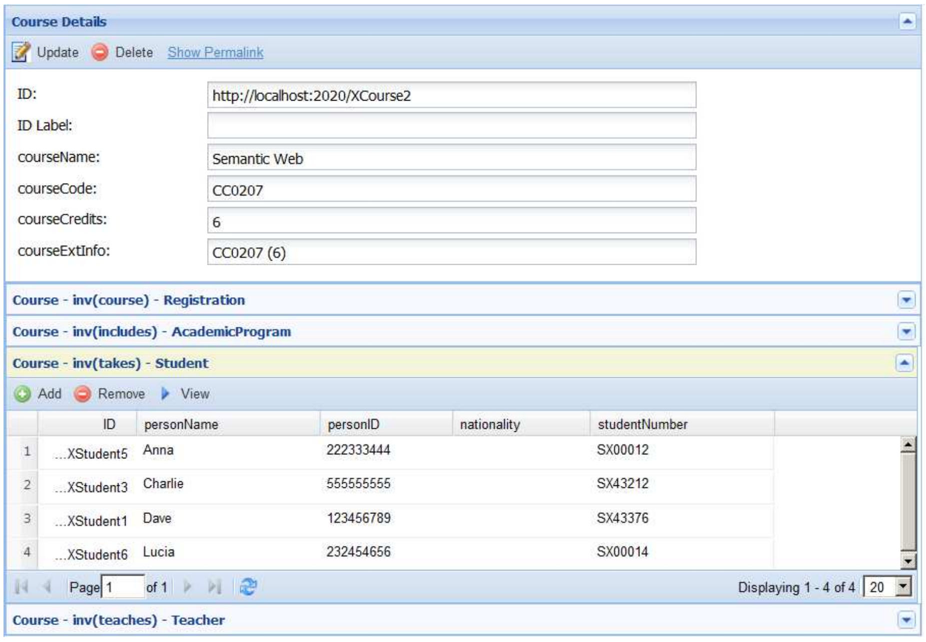
	ID	programName
1	...Program1	Computer Science

	ID	personName	personID	nationality	studentName	studentNumber
1	...XStudent1	Dave	123456789		Dave	SX43376
2	...XStudent3	Charlie	555555555		Charlie	SX43212
3	...XStudent5	Anna	222333444		Anna	SX00012
4	...XStudent6	Lucia	232454656		Lucia	SX00014

Fig. 3. The related item view options for the instance list in OBIS

Further ontology data exploration options include the sorting and filtering, export to MS Excel as well as related data item view allowing the user to select one or several instance links defined for the class in the ontology as well as a list of instances and obtain the lists of items connected to the selected ontology instances via the specified links. Figure 3 demonstrate this facility for the courses 'Semantic Web' and 'Quantum Computations' in the mini-University example (note the properties *includes* and *takes* coming into the *Course* class in the data ontology in Figure 1).

There is also a detailed view form for instances with fields for attribute values and links in the ontology data coming in (the direct links) and going out (the inverse link), as in Figure 4. Show Permalink option helps sharing the instance property page among various developers working with the same OBIS server application.


Course Details

Update Delete Show Permalink

ID:	http://localhost:2020/XCourse2
ID Label:	
courseName:	Semantic Web
courseCode:	CC0207
courseCredits:	6
courseExtInfo:	CC0207 (6)

Course - inv(course) - Registration

Course - inv(includes) - AcademicProgram

Course - inv(takes) - Student

Add Remove View

	ID	personName	personID	nationality	studentNumber
1	...XStudent5	Anna	222333444		SX00012
2	...XStudent3	Charlie	555555555		SX43212
3	...XStudent1	Dave	123456789		SX43376
4	...XStudent6	Lucia	232454656		SX00014

Page 1 of 1 Displaying 1 - 4 of 4 20

Course - inv(teaches) - Teacher

Fig. 4. A class instance view, with link options.

OBIS can be used not only as the ontology data information viewer. In the case, if the underlying RDF data store supports SPARQL data updates, there is possibility of new information entry, as well as information update and delete in OBIS, as well. The class instance information entry and edit forms are similar to instance view forms of Figure 4 (with information save option enabled); the link information is updated via the *Add* option in the tables showing link relations.

Further options in OBIS are report creation that allows also execution of textual SPARQL queries against the underlying RDF data store as well as form and report information export to MS Excel. The internal report definition tool in OBIS is work in progress. To ease writing textual SPARQL queries one can use e.g. a ViziQuer tool[4] [17], among the others.

3 Enhancing OBIS Applications

An information system user interface that is based just on the reproduction of the structure of ontology classes and data and object properties (with an analogy with RDB table, attribute and link structure) may appear to be too simplistic for a useful information system. One could desire e.g. more specific table and form view definitions for particular classes. In particular, the classes containing the classifier data, or, in general, the links among classes with their target maximum cardinality not exceeding 1 (i.e., the n:1 links) may benefit from special visual representation in the user interface. There are three basic mechanisms of enhancing OBIS applications:

- specification of cardinality information in the ontology (this requires using the standard means available in OWL, so the effects of cardinality information availability can be considered basic OBIS application generation aspect);
- extending OWL ontology with specific user interface annotations, and
- customizing (configuring) the application after its initial generation.

The main user interface annotations, supported by OBIS, are the following[5]:

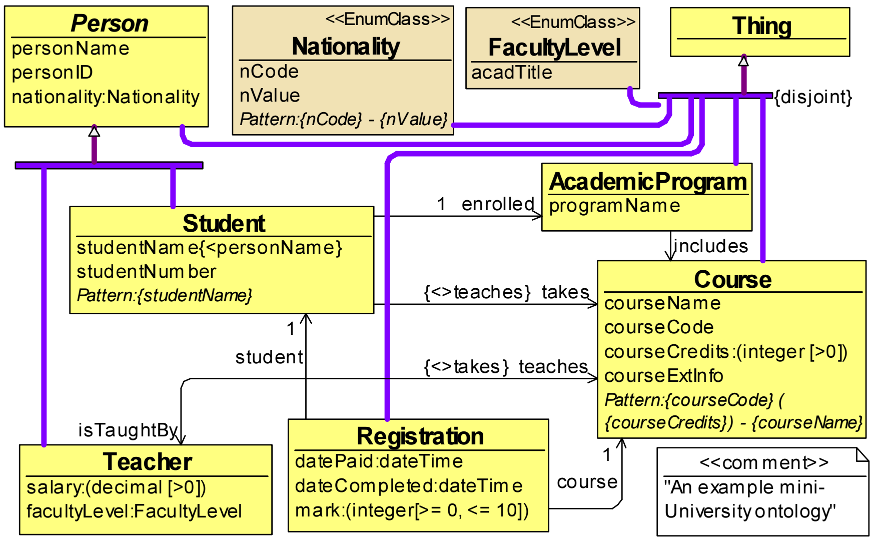
- *obis:textPattern.* Defines textual form for class instance presentation via relations with maximum cardinality 1. The textual form of linked instances of such classes are included into linking instance presentations both in table and form and edit views (a combo-box is used for the edit view).
- *obis:isEnumerated.* Specifies a class to be a classifier class (an enumerated class), so that a textual form of its instance presentation is available disregarding an explicit *obis:textPattern* specification by the user.
- *obis:defaultOrder*. Specifies the default order of attributes (both data attributes and classifier attributes) for class instances within a class table or form view. The OWL ontology axiom structure does not foresee a possibility to specify the order of the attributes for a class. The *obis:defaultOrder* annotations create a compensating mechanism. An automated generation of these of axioms is built in OWLGrEd editor extension OWLGrEd/OBIS [14,18] for data ontology handling. So, with appropriate ontology editor support an IS developer would not notice the existence of these annotations.
- *obis:isAbstract*. Specifies that a class is abstract and that a possibility to create new instances as elements of this class is not to be included in UI.
- *obis:isDerived*. Specifies that a data or object property is derived and that a possibility to create new values or links corresponding to this property is not to be included in UI.
- *obis:view*. Specifies the presence and order of attributes (both data attributes and classifier attributes) shown for class instances within a table or form view. May include both attributes ascribed for the class and its subclasses in the

[4] Can be downloaded from viziquer.lumii.lv

[5] The ontology prefix obis: stands for the namespace http://obis.lumii.lv/2013/01/obis#

ontology, as well as any other attributes (e.g. attributes whose domain (ascription) in the ontology is not defined). The *obis:view* annotations, where specified, take precedence over *obis:defaultOrder* annotations and it is in general considered that the *obis:view* annotations are to be entered manualy.

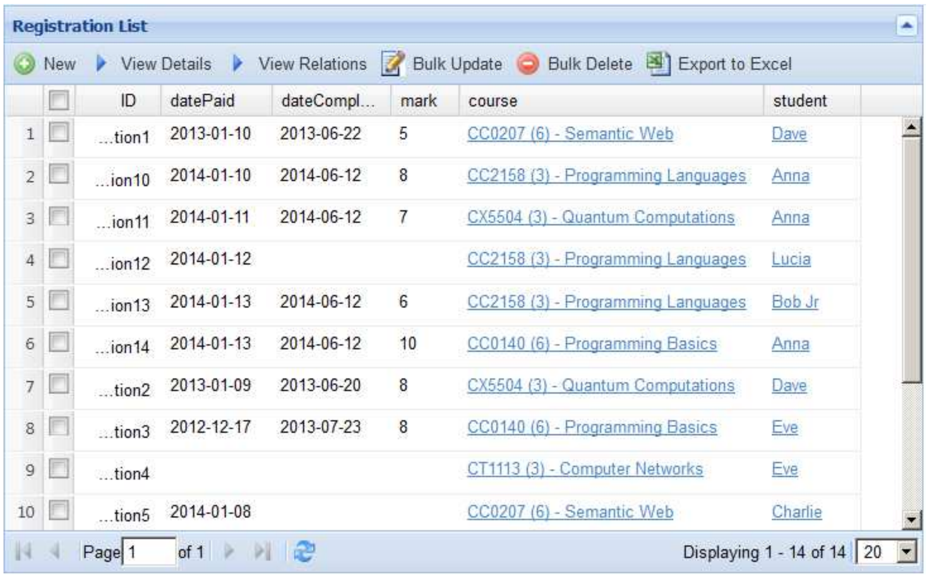
Figure 5 shows the mini-University ontology of Figure 1 extended with some example user interface annotations added. The notation used is domain specific ontology notation in the OWLGrEd ontology editor [18], it presents the *obis:textPattern* annotation in a textual form, *obis:isEnumerated* as <<EnumClass>> stereotype and *obis:isAbstract* as italics font face of the class name. The *obis:defaultOrder* annotations are created automatically by the OWLGrEd/OBIS editor and are not seen by the ontology end user.



```
AnnotationAssertion(obis:defaultOrder :Nationality "nCode,nValue")
AnnotationAssertion(obis:isEnumerated :Nationality "true")
AnnotationAssertion(obis:textPattern :Nationality "{nCode} - {nValue}")
AnnotationAssertion(obis:isAbstract :Person "true")
AnnotationAssertion(obis:defaultOrder :Person "personName,personID,nationality")
```

Fig. 5. Mini-University ontology with obis: annotations and annotation examples in OWL Functional Syntax

Figure 6 shows the effect of the introduced text pattern annotations on the presentation of Registration class in the generated OBIS application. Note that without the "linking in" the course and student information, the presentation of the Registration table would appear rather non-informative. The links behind the instances in the table open the detail views of the corresponding referenced instances. Notice that the generation of the illustrated user interface has been possible with just a minor manual tuning of the ontology (introducing *obis:textPattern* annotations to the *Student* and *Course* classes).



	ID	datePaid	dateCompl...	mark	course	student
1	...tion1	2013-01-10	2013-06-22	5	CC0207 (6) - Semantic Web	Dave
2	...ion10	2014-01-10	2014-06-12	8	CC2158 (3) - Programming Languages	Anna
3	...ion11	2014-01-11	2014-06-12	7	CX5504 (3) - Quantum Computations	Anna
4	...ion12	2014-01-12			CC2158 (3) - Programming Languages	Lucia
5	...ion13	2014-01-13	2014-06-12	6	CC2158 (3) - Programming Languages	Bob Jr
6	...ion14	2014-01-13	2014-06-12	10	CC0140 (6) - Programming Basics	Anna
7	...tion2	2013-01-09	2013-06-20	8	CX5504 (3) - Quantum Computations	Dave
8	...tion3	2012-12-17	2013-07-23	8	CC0140 (6) - Programming Basics	Eve
9	...tion4				CT1113 (3) - Computer Networks	Eve
10	...tion5	2014-01-08			CC0207 (6) - Semantic Web	Charlie

Fig. 6. A class view including the linked instance descriptions.

The option of fine-tuning the OBIS application after its initial generation is based on the principle of availability of the OBIS application configuration also as OBIS application itself, so OBIS itself can be used to perform its application fine-tuning.

4 Discussion and Conclusions

The OBIS Framework demonstrates the feasibility of automated generation of feature-rich information systems just from annotated data ontologies. Besides a potential stand-alone usability for viewing or managing data structured in accordance to the RDF format, the concepts from OBIS might appear useful in the context of other editors of instances corresponding to structure defined by OWL ontology or RDF Schema, such as Web Protégé [19] or TopBraid Ensemble [20] that aim at similar ontology instance editing form functionality.

Although the OBIS tool can be used to interpret data ontologies created by different tools, its integration within the OWLGrEd tool [16] and its OBIS-extension [18], [14] may appear beneficial for the ease of information system customization in OBIS. We note also an important use case of OBIS in viewing RDF data corresponding to conceptual models of data coming from relational databases [14].

There is work in progress towards creating the custom-built reports, extending the validation rule set, as well as extending the set of annotations, interpretable by OBIS, however, trying to keep the annotations to be specified by the user to a comprehensible minimum. The advances towards creating user-comprehensible UI annotations that allow generation of fully functional UI from the annotated data model could be of value also for UI generation from UML-style conceptual models.

Another possible venue of the OBIS extension is via integration of the visual SPARQL query creation tool ViziQuer [17], however, this option requires a technological re-work of the ViziQuer tool via placing it into web-based framework.

References

1. Resource Description Framework (RDF), http://www.w3.org/RDF/
2. RDF Schema [WWW] http://www.w3.org/TR/rdf-schema/
3. Motik, B; Patel-Schneider P.F; Parsia B.: OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax, 2009
4. SPARQL 1.1 Overview. W3C Recommendation 21 March 2013 [WWW] http://www.w3.org/TR/sparql11-overview/
5. Linked Data, http://linkeddata.org
6. Tim Berners-Lee, James Hendler and Ora Lassila, "The Semantic Web", Scientific American, May 2001, p. 29-37.
7. R2RML: RDB to RDF Mapping Language [WWW] http://www.w3.org/TR/r2rml/
8. C.Blakeley: "RDF Views of SQL Data (Declarative SQL Schema to RDF Mapping)", OpenLink Software, 2007.
9. D2RQ Platform. Treating Non-RDF Relational Databases as Virtual RDF Graphs. http://www4.wiwiss.fu-berlin.de/bizer/D2RQ/spec/
10. Bagosi, T., Calvanese, D., Hardi, J., Komla-Ebri, S., Lanti, D., Rezk, M., Rodriguez-Muro, M., Slusnys, M., & Xiao, G. (2014). The Ontop framework for ontology based data access. In Zhao, D., Du, J., Wang, H., Wang, P., Ji, D., & Pan, J. Z. (Eds.), CSWS 2014, Vol. 480 of Communications in Computer and Information Science, pp. 67-77. Springer.
11. K.Čerāns, G.Būmans, RDB2OWL: a RDB-to-RDF/OWL Mapping Specification Language // J.Barzdins and M.Kirikova (eds.), Databases and Information Systems VI, IOS Press 2011, p.139-152.
12. Stardog, http://stardog.com/
13. G.Barzdins, E.Liepins, M.Veilande, M.Zviedris: Semantic Latvia Approach in the Medical Domain. // Proc. 8th International Baltic Conference on Databases and Information Systems. H.M.Haav, A.Kalja (eds.), TUT Press, pp. 89-102. (2008).
14. K. Cerans, G. Barzdins, G. Bumans, J. Ovcinnikova, S. Rikacovs, A. Romane and M. Zviedris. A Relational Database Semantic Re-Engineering Technology and Tools // Baltic Journal of Modern Computing (BJMC), Vol. 3 (2014), No. 3, pp. 183-198.
15. M.Zviedris, A.Romane, G.Barzdins, K.Cerans. Ontology-Based Information System // Proceedings of JIST'2013, LNCS, Vol. 8388, 2014, pp. 33-47.
16. Barzdins, J.; Barzdins, G.; Cerans, K.; Liepins, R.; Sprogis, A.: OWLGrEd: a UML Style Graphical Notation and Editor for OWL 2. In Proc. of OWLED 2010, 2010.
17. Zviedris M., Barzdins G. (2011), ViziQuer: A Tool to Explore and Query SPARQL Endpoints // The Semantic Web: Research and Applications, LNCS, 2011, Volume 6644/2011, pp. 441-445
18. K.Čerāns, J.Ovčiņņikova, R.Liepiņš, A.Sproģis, Advanced OWL 2.0 Ontology Visualization in OWLGrEd // A.Caplinskas, G.Dzemyda, A.Lupeikiene, O.Vasilecas (eds.), Databases and Information Systems VII, IOS Press, Frontiers in Artificial Intelligence and Applications, Vol 249, 2013, pp.41-54
19. WebProtege, http://webprotege.stanford.edu
20. TopBraid Ensemble, http://www.topquadrant.com/product/TB_Ensemble.html