ISSN 1407-7493 COMPUTER SCIENCE DATORZINĀTNE 2008-34

APPLIED COMPUTER SYSTEMS
LIETIŠĶĀS DATORSISTĒMAS

# OWL ONTOLOGY TRANSFORMATION INTO CONCEPT MAP

# OWL ONTOLOĢIJU TRANSFORMĀCIJA JĒDZIENU TĪKLOS

**Vita Graudina**, Riga Technical University, 1 Kalku str., Riga, LV-1658, researcher, M.sc.ing.,
vita.graudina@cs.rtu.lv

***Ontology, concept map, transformation***

## 1. Introduction

During last two decades usage of ontologies in the field of computer science has been rapidly increased. First attempts of using ontologies were in subfields of artificial intelligence related to researches in knowledge engineering, natural language processing and knowledge representation. At the end of 1990s ontologies started to propagate in other subfields, like intelligent information integration, information retrieval from the Internet, knowledge management, later also in e-commerce, Semantic Web and other. The most important step in ontology evolution is development of in 2004 [1]. Development of Web Ontology Language (traditionally, abbreviated as OWL) together with many tools for ontology construction (e.g., Protégé, WebODE etc.) made ontologies quite widespread. In the Internet there are more than 33 000 ontologies encoded in OWL (searched using www.google.com in September, 2007).

Taking into account obvious similarities between ontologies and concept maps the research of ontology transformation into concept map is undertaken. Concept map generation from existing OWL ontologies could reduce workload of teachers who use concept maps in teaching process, for example, for knowledge assessment. The transformation offers to teachers an initial concept map created automatically, and he/she only needs to refine it according to their requirements, extending or narrowing it.

This paper describes the initial phase of the research where corresponding elements of ontology and concept map are identified. The remaining of the paper is organized as following: in Section 2 definitions and clarification of terms and concepts used in this paper is given. Then in Section 3 mappings between OWL language constructs and elements of concept map are demonstrated. In Section 4 the experiment of applying proposed mappings is

described. Section 5 is dedicated to related work in OWL ontology transformation, and finally, future work and some conclusions are outlined.

## 2. Background

In this chapter main terms used in this paper are clarified and definitions are given:

- *Concept maps* are graphs, which include concepts as nodes and relations between them as arcs. Sometimes so called linking phrases are used. Usually, concept maps are represented as hierarchies with most general concepts at the top of the map and more specific concepts are placed at the lowest levels [2]. Concept maps can have different topologies, such as linear, circular, hub/spoke, tree, network/net [3].
- *An ontology* after one of its definition [4] "is a formal explicit description of concepts in a domain of discourse (classes (sometimes called concepts)), properties of each concept describing various features and attributes of the concept (slots (sometimes called roles or properties)), and restrictions on slots (facets (sometimes called role restrictions))". Ontologies can be different according to their expressiveness, beginning with simple vocabularies and taxonomies, ending with ontologies containing formal restrictions and constraints [5].
- *Web Ontology Language (OWL)* is ontology language, used to define ontology for particular domain. OWL ontology is a set of axioms describing classes, properties, and relationships between them. OWL language provides more features than RDFS (Resource Description Framework Schema) for defining ontology elements. OWL ontologies use RDF/XML (Resource Description Framework)/(eXtended Markup Language) syntax [6].

## 3. Mapping between OWL ontology elements and concept map elements

Based on analysis of OWL language syntax [1, 6, 7] and already existing OWL ontologies corresponding elements of OWL ontology and concept map are gained. In the Table 1 the results of analysis are given. The table contains elements of OWL and corresponding elements of concept map. The OWL elements which are not related with a concept map are omitted, e.g., cardinality restrictions, which don't have impact to the concept map structure, OWL constructs such as `owl:versionInfo` which describes the version of the particular ontology, `owl:backwardCompatibleWith` which contains reference to another ontology identifying the specified ontology as a prior version of the containing ontology, and further indicates that it is backward compatible with it which is prior version, etc. The elements, which are not unambiguous understandable without examples, are also omitted in the Table 1. They are described with examples after the table.

**Table 1. OWL elements with corresponding concept map elements**

| # | OWL element | Concept map element |
|---|---|---|
| 1. | `owl:AllDifferent` | Concept |
| 2. | `owl:allValuesFrom` | Concept |
| 3. | `owl:Class` | Concept |
| 4. | `owl:complementOf` | Concept |
| 5. | `owl:DataRange` | Concept |
| 6. | `owl:DatatypeProperty` | Concept |

| # | OWL element | Concept map element |
|---|---|---|
| 7. | owl:DeprecatedClass | Concept |
| 8. | owl:DeprecatedProperty | Link |
| 9. | owl:differentFrom | Concept |
| 10. | owl:disjointwith | Concept |
| 11. | owl:distinctMembers | Concept |
| 12. | owl:equivalentClass | Concept |
| 13. | owl:equivalentProperty | Link |
| 14. | owl:FunctionalProperty | Concept |
| 15. | owl:intersectionOf | Concept |
| 16. | owl:inverseOf | Link |
| 17. | owl:inverseFunctionalProperty | Concept |
| 18. | owl:Nothing | Concept |
| 19. | owl:ObjectProperty | Link |
| 20. | owl:oneOf | Concept |
| 21. | owl:someValuesFrom | Concept |
| 22. | owl:SymmetricProperty | Link |
| 23. | owl:Thing | Concept |
| 24. | owl:TransitiveProperty | Link |
| 25. | owl:unionOf | Concept |

With mappings between OWL elements and corresponding elements of concept maps are not enough to implement software for ontology transformation into concept map. It is not enough because in OWL ontology there can be used also RDF and RDFS elements, as well as the same elements of ontology can be described in different ways. Therefore further in this section several listings of OWL code, comments about them, and corresponding concept maps are given. In the Figure 1 basic elements of OWL listing are shown. Each element of the ontology is described using XML syntax. The type of the element is described with the XML element name, for example, `owl:Class`, `owl:DatatypeProperty`, `owl:ObjectProperty` etc. After the XML element name follows the identifier of the element given by the user. In the XML element name tag additional information about the element can be placed, like, comments, hierarchal information, restrictions, cardinality etc.

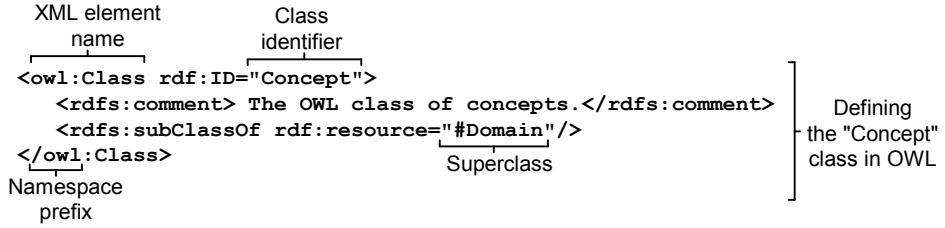



**Figure 1. Example of OWL syntax**

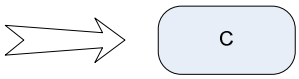
The identified mappings between OWL ontology and concept map are listed below. The listings of OWL code are followed by graphical representation of corresponding concept map element or elements.

- Class “C” is defined.

```
<owl:Class rdf:ID=”C”>
</owl:Class>
```

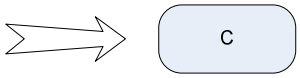
----------or----------

```
<rdf:Description rdf:about="#C">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Class"/>
</rdf:Description>
```



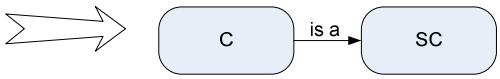

- Reference to class “C”.

```
<owl:Class rdf:about=”#C”>
</owl:Class>
```




- Class “SC” is superclass of class “C”. The link between class and superclass is oriented and labelled with “is a”.

```
<owl:Class rdf:ID="C">
    <rdfs:subClassOf rdf:resource="#SC"/>
</owl:Class>
```




----------or----------

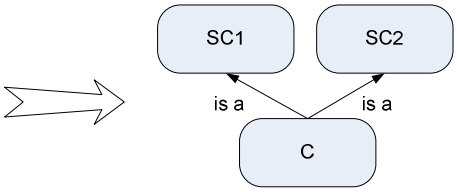
```
<owl:Class rdf:ID="C">
    <rdfs:subClassOf>
        <owl:Class rdf:ID="SC"/>
    </rdfs:subClassOf>
</owl:Class>
```

----------or----------

```
<owl:Class rdf:ID="C">
    <rdfs:subClassOf>
        <owl:Class rdf:about="#SC"/>
    </rdfs:subClassOf>
</owl:Class>
```

- Class “C” has two superclasses “SC1” and “SC2”.

```
<owl:Class rdf:ID="J">
    <rdfs:subClassOf rdf:resource="#VK1"/>
    <rdfs:subClassOf rdf:resource="#VK2"/>
</owl:Class>
```

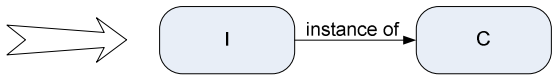



- “I” is instance of class “C”. The link between instance and class is oriented and labelled with “instance of”.

```
<C rdf:ID="I"/>
```

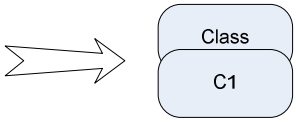
----------or----------

```
<C rdf:ID="I">
</C>
```



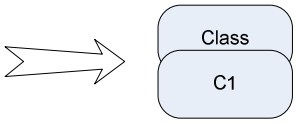

- Alternative name or human understandable name of class “C1” is “Class”. Thus the synonyms can be defined. If the `rdfs:label` is replaced with `rdfs:comment` then more descriptive text can be added as well as link to external source or file can be attached.

```
<owl:Class rdf:ID="C1">
    <rdfs:label>Class</rdfs:label>
</owl:Class>
```



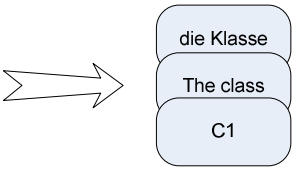
- Alternative name of the class “C1” is “Class” and datatype for the label is defined.

```
<owl:Class rdf:ID="C1">
    <rdfs:label
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">
Class</rdfs:label>
</owl:Class>
```




- The class name in different languages.

```
<owl:Class rdf:ID="C1">
    <rdfs:label xml:lang="en">The class</rdfs:label>
    <rdfs:label xml:lang="de">die Klasse</rdfs:label>
</owl:Class>
```

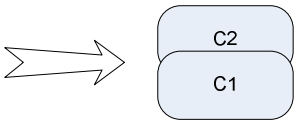



- Classes “C1” and “C2” are equivalent or the same – synonyms.

```
<owl:Class rdf:ID="C1">
    <owl:equivalentClass>
        <owl:Class rdf:ID="C2"/>
    </owl:equivalentClass>
</owl:Class>
```

----------or----------

```
<owl:Class rdf:ID="C1">
    <owl:sameAs rdf:resource="#C2"/>
</owl:Class>
```

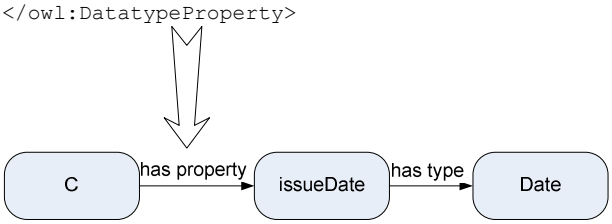



- The relationship “links” between classes “From” and “To”. In OWL language semantic relationship between two classes or instances is called object property.

```
<owl:ObjectProperty rdf:ID="links">
    <rdfs:domain rdf:resource="#From"/>
    <rdfs:range rdf:resource="#To"/>
</owl:ObjectProperty>
```

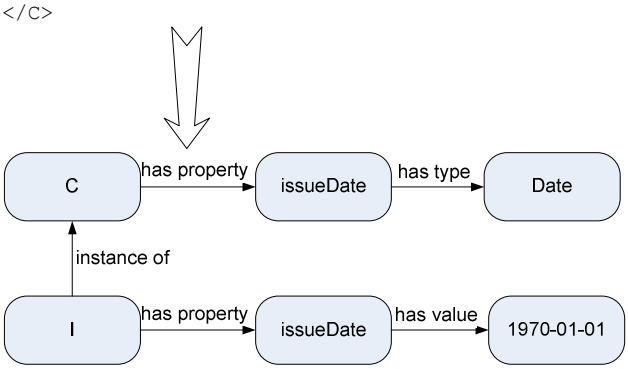

- Class “C” has property “issueDate” with defined datatype Date. In OWL language the relationship between class/instance and datatype (build-in XML Schema datatype) is called datatype property. In concept map datatype property is represented as the concept with the link “has property” to the concept which is characterized by it, and the property concept is linked to the datataype concept with the link “has type”.

```
<owl:DatatypeProperty rdf:ID="issueDate">
    <rdfs:domain rdf:resource="#C"/>
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
</owl:DatatypeProperty>
```



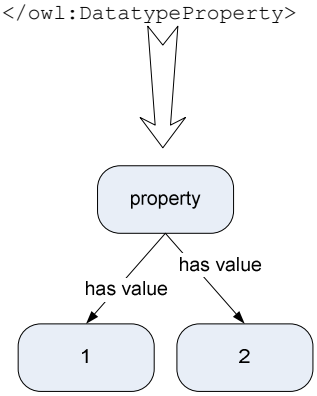

- “I” is instance of class “C” (see, previous paragraph) and property’s “issueDate” value is specified as “1970-01-01”. In concept map the link between property and its value is labelled with “has value”.

```
<C rdf:ID="I">
    <issueDate rdf:datatype="http://www.w3.org/2001/XMLSchema#date">
        1970-01-01
    </issueDate>
</C>
```




- Datatype properties can have predefined values.

```
<owl:DatatypeProperty rdf:ID="property">
    <rdfs:range>
      <owl:DataRange>
        <owl:oneOf rdf:parseType="Resource">
          <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
          >1</rdf:first>
          <rdf:rest rdf:parseType="Resource">
            <rdf:rest rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#nil"/>
            <rdf:first rdf:datatype="http://www.w3.org/2001/XMLSchema#int"
            >2</rdf:first>
          </rdf:rest>
        </owl:oneOf>
      </owl:DataRange>
    </rdfs:range>
</owl:DatatypeProperty>
```

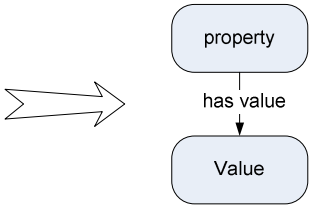



- Property has restriction on value range.

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#property"/>
  <owl:allValuesFrom rdf:resource="#Value"/>
</owl:Restriction>
```

----------or----------

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#property" />
  <owl:someValuesFrom rdf:resource="#Value"/>
</owl:Restriction>
```




----------or----------

```
<owl:Restriction>
  <owl:onProperty rdf:resource="#property"/>
  <owl:hasValue rdf:resource="#Value"/>
</owl:Restriction>
```

- Object property as well as datatype property can be functional properties, but it doesn't impact representation of concept map.

```
<owl:FunctionalProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:FunctionalProperty>
```

----------or----------

```
<owl:FunctionalProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#DatatypeProperty"/>
</owl:FunctionalProperty>
```

- Inverse functional property can be only object property.

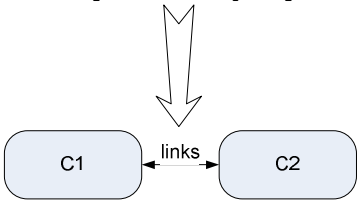
```
<owl:InverseFunctionalProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:InverseFunctionalProperty>
```

- Inverse property describes object property with opposite direction.

```
<owl:ObjectProperty rdf:about="#isLinked">
    <owl:inverseOf rdf:resource="#links"/>
</owl:ObjectProperty>
```


- Symmetric property describes object property where link is bidirectional.

```
<owl:SymmetricProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
    <rdfs:domain rdf:resource="#C1"/>
    <rdfs:range rdf:resource="#C2"/>
</owl:SymmetricProperty>
```




- Transitive property describes object property. Property "links" links class "C1" with class "C2" and class "C2" with class "C3", and as "links" is transitive property then also "C1" "links" "C3".

```
<owl:Class rdf:ID="C1">
    <links>
      <owl:Class rdf:ID="C2"/>
    </links>
</owl:Class>

<owl:Class rdf:ID="C3"/>

<owl:Class rdf:about="#C2">
    <links rdf:resource="#C3"/>
</owl:Class>
<owl:TransitiveProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

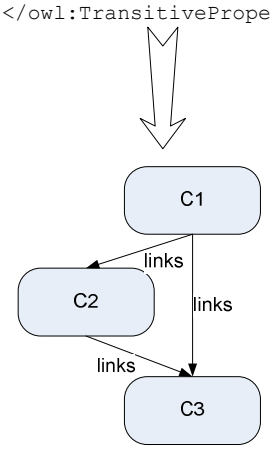
----------or----------

```
<owl:Class rdf:ID="C1">
    <links rdf:resource="#C2"/>
</owl:Class>

<owl:Class rdf:ID="C3"/>

<owl:Class rdf:ID="C2">
    <links rdf:resource="#C3"/>
</owl:Class>

<owl:TransitiveProperty rdf:ID="links">
    <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#ObjectProperty"/>
</owl:TransitiveProperty>
```

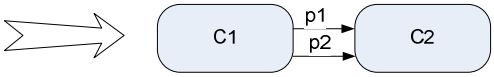



- Properties “p1” and “p2” are equivalent or the same– synonyms.

```
<owl:ObjectProperty rdf:ID="p1">
    <owl:equivalentProperty rdf:resource="#p2"/>
</owl:ObjectProperty>
```

----------or----------

```
<owl:ObjectProperty rdf:ID="p1">
    <owl:sameAs rdf:resource="#p2"/>
</owl:ObjectProperty>
```




## 4. Laboratory experiment

The laboratory experiment was done to check appropriateness of proposed assumptions about OWL elements correspondence to elements of concept maps. Ten freely at Internet available ontologies encoded in OWL were chosen for the experiment (see, Table 2). All ontologies were related to computer science and information technologies. Swoogle Semantic Web search engine [8] developed by Google was used to search ontologies. This search engine was chosen because it searches within ontologies to find specified keywords as classes of

ontology, not only simple text matching. Using other search engines there were found ontologies were specified keywords are in comments and in other file describing parts.

**Table 2. Characteristics of ontologies used in case study**

| # | Ontology's web site | Topic | Number of | | | |
|---|---|---|---|---|---|---|
| | | | File size (kb) | Classes | Properties | Instances |
| 1. | http://didaskon.corrib.org/download/ARSI06_19_LOO_2006_07_20_FC.owl | Learning objects | 39 | 50 | 70 | 27 |
| 2. | http://ontology.ihmc.us/Computing/Computing Entity.owl | Computer | 5 | 9 | 7 | - |
| 3. | http://geobrain.laits.gmu.edu/ontology/2004/03/fgdc-RS_extension-data_quality.owl | Data quality | 54 | 19 | 59 | - |
| 4. | http://metacognition.info/ontologies/FtssConfig.owl | 4Suite repository | 16 | 7 | 26 | - |
| 5. | http://monet.nag.co.uk/cocoon/monet/publicdocs/ontologies/hardware.owl | Hardware | 5 | 21 | 4 | - |
| 6. | http://www.ksl.stanford.edu/DAML/laptops.owl | Laptops | 32 | 42 | 36 | 12 |
| 7. | http://bw.rulez-forever.com/onto/mdom.owl | Artificial intelligence | 51 | 30 | 22 | 143 |
| 8 | http://www.atl.lmco.com/projects/ontology/ontologies/network/networkA.owl | Networks | 5 | 27 | 5 | - |
| 9. | http://www.itee.uq.edu.au/~dwood/ontologies/sec.owl | Software engineering | 21 | 13 | 29 | - |
| 10. | http://monet.nag.co.uk/cocoon/monet/publicdocs/ontologies/software.owl | Programming languages | 11 | 22 | 4 | - |

In the experiment there was made manual transformation from chosen ontologies to concept maps using identified mapping between ontology code elements and concept map elements. For example concept map generated from Network Ontology the ontology #8 is shown in Figure 2.

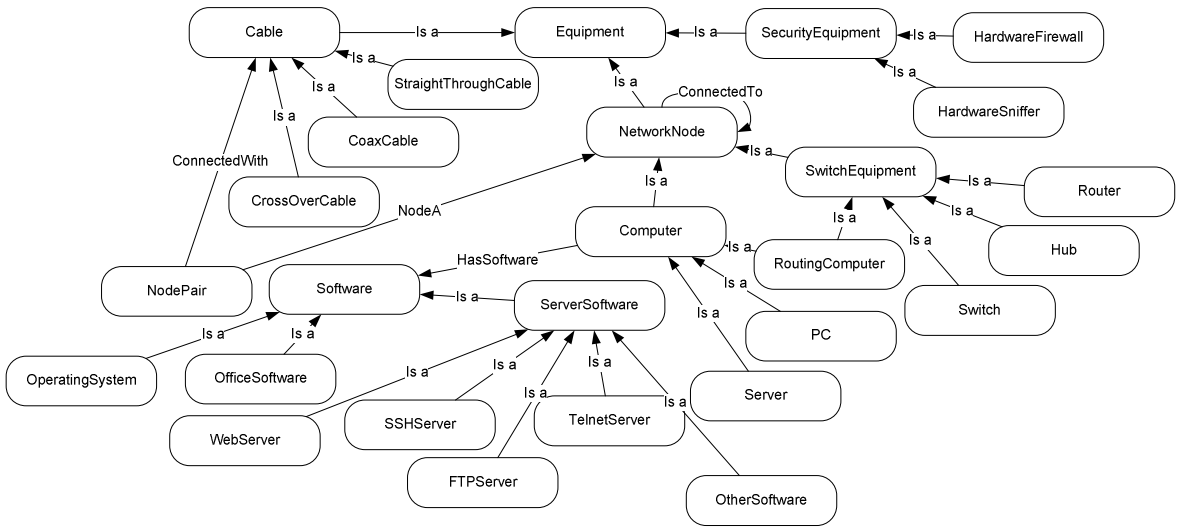


**Figure 2. Concept map generated from Network Ontology**

Conclusions from derived concept maps are following:

- The proposed mapping principles are successfully applicable for concept map generation from ontology.
- The proposed mappings should be refined with additional mappings for such OWL constructs as `owl:Restriction`, `owl:differentFrom`, because these constructions are used in ontology to extended class definitions.

Concept map representation problems have appeared in such cases:

- if the ontology has Boolean operations on classes (`owl:unionOf`, `owl:intersectionOf`, `owl:complementOf`) to define new class;
- if the ontology has a hierarchy of properties;
- if the ontology has several namespaces defined or imports other ontology, the problem occurs in concept naming (`<owl:Class rdf:about=http://owl.protege. stanford.edu#Wireless-Network-Card/>`).

All emerged problems can be solved by defining additional mappings OWL ontology to concept maps, and appropriate constraints and conditions to their usage.

## 5. Related work

Study of literature related to OWL ontology transformation showed that the basic technology for ontology transformation to other formats is usage of metamodels. There already exist metamodels for mapping from UML (Unified Modelling Language) to OWL, from OWL to UML, from OWL to Topic Maps, from Topics Maps to OWL, from OWL to Common Logic within Meta-Object Facility Query/View/Transformation framework [9, 10]. Transformation from UML to OWL can be done using XSLT (Extensible Stylesheet Language Transformations) document which contains transformation rules [10] or Description Logics [11]. Transformation from OWL to UML is also supported by software [12]. Other ontology transformations are related to enterprise ontology transformation to conceptual data model [13, 14], business rules [15] and relational database [16].

## 6. Future work

Future work is related to implementation of proposed transformation into a software application. The closest tasks are to refine current informal mapping between ontology and concept map, to make formal transformation algorithm and to define metamodels for clearness and convenience of transformation.

At the beginning of implementation the author should choose application programming interface (API) developed to access OWL ontologies. Currently, there exist three APIs for OWL, i.e., Jena [17], OWL API [18] and Protégé OWL-API [19]. Actually, the author should choose between Jena and Protégé OWL-API, because they have better documentation and are more often used than OWL API. At the present moment it seems that Jena is more suitable not only because it is used in Protégé OWL-API and ontology construction tool Protégé to access reasoners, but also it is applicable to DAML (DARPA Agent Markup Language) and

RDFS ontology languages. More studies of literature and practical experiments are needed to make final decision.

Further tasks are connected with software implementation and testing, as well as integration in already developed knowledge assessment system based on concept maps [].

## 7. Conclusions

Defined mappings between OWL ontology elements and elements of the concept map were checked applying them manually to existing ontologies. During applying proposed mappings there were found some incompleteness mainly connected with usage of namespaces and imports of other ontologies. Manual transformation showed that there are problems with representation of concept maps if Boolean operations to OWL classes are used to define other classes. The same problems occurred at representation of inheritance and a hierarchy of properties. Discovered drawbacks can be solved by additional mappings and conditions on their usage. For clearness of mappings metamodels of concept maps and OWL ontologies should be defined.

The concept maps derived from ontologies proved quite good, therefore they or parts of them can be used as teachers' concept maps in already developed knowledge assessment system based on concept maps [20].

## 8. Acknowledgement

This work has been partly supported by the European Social Fund within the National Program "Support for the carrying out doctoral study program's and post-doctoral researches" project "Support for the development of doctoral studies at Riga Technical University".

The main results are outcomes of the project R7197 "Development of an ontology-based intelligent system for task generation in the form of concept maps and knowledge assessment".

## 9. References

1. OWL Web Ontology Language Reference. Available online (last visited 06.07.2007): http://www.w3.org/TR/2004/REC-owl-ref-20040210/
2. Novak J.D., Canas A.J. The theory underlying concept maps and how to construct them. Technical Report IHCM CmapTools 2006-1, 2006.
3. Yin Y., Vanides J., Ruiz-Primo M.A., Ayala C.C., Shavelson R.J. Comparison of two concept-mapping techniques: implications for scoring, interpretation, and use // In: Journal of Research in Science Teaching Vol. 42, No. 2, 2005, p 166-184.
4. Noy N.F., McGuinness D.L. Ontology Development 101: A Guide to Creating Your First Ontology. Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880, March 2001. Available online (last visited 28.09.2007): http://ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness.pdf
5. Lassila O., McGuinness D. The Role of Frame–Based Representation on the Semantic Web // In: Electronic Transactions on Artificial Intelligence, Vol. 5, 2001.
6. Lacy L.W. OWL: Representing Information Using the Web Ontology Language. Trafford Publishing, 2005.
7. OWL Web Ontology Language Guide. Available online (last visited 06.07.2007): http://www.w3.org/TR/2004/REC-owl-guide-20040210/

8. Swoogle Semantic Web Search Engine. Available online (last visited 16.08.2007): http://swoogle.umbc.edu/
9. OMG. Ontology Definition Metamodel Specification. Available online (last visited 06.09.2007): http://omg.org/docs/ptc/06-10-11.pdf
10. Na H-S., Choi O-H, Lim J-E. A Method for Building Domain Ontologies based on the Transformation of UML Models // In: Proceedings of Fourth International Conference on Software Engineering Research, Management and Applications (SERA'06), 2006, p. 332-338.
11. Colomb R.M., Gerber A., Lawley M. Issues in Mapping Metamodels in the Ontology Development Metamodel Using QVT // In: The 1st International Workshop on the Model-Driven Semantic Web, 2004.
12. UMLBackendMapping. Available online (last visited 06.09.2007): http://protege.cim3.net/cgi-bin/wiki.pl?UMLBackendMapping
13. Vasilecas O., Bugaite D., Trinkunas J. On Approach for Enterprise Ontology Transformation into Conceptual Model // In: Proceedings of International Conference on Computer Systems and Technologies - CompSysTech'06, 2006, p. IIIA.23-1 - IIIA.23-6.
14. Trinkunas J., Vasilecas O. A Graph oriented model for ontology transformation into conceptual data model // In: Information Technology and Control. Vol. 36, No.1A, 2007, p. 126-132.
15. Bugaite D., Vasilecas O. Framework on application domain ontology transformation into set of business rules // In: Proceedings of International Conference on Computer Systems and Technologies - CompSysTech'05, 2005, p. IIIB.8-1 - IIIB.8-6.
16. Vysniauskas E., Nemuraite L. Transforming Ontology Representation from OWL to Relational Database. // In: Information Technology and Control, Vol. 35A, No. 3, 2006, p. 333-343.
17. Jena – A Semantic Web Framework for Java. Available online (last visited 29.09.2007): http://jena.sourceforge.net/
18. An API for OWL. Available online (last visited 26.09.2007): http://owl.man.ac.uk/api/readme.html
19. Protégé-OWL API programmer's guide. Available online (last visited 26.09.2007): http://protege.stanford.edu/plugins/owl/api/guide.html
20. Anohina A., Pozdnakovs D., Grundspenkis J. Changing the Degree of Task Difficulty in Concept Map Based Assessment System // In: Proceedings of the IADIS International Conference "e-Learning 2007" p. 443-450.

**Graudiņa V. OWL ontoloģiju transformācija jēdzienu tīklos**

*Šis raksts ir veltīts ontoloģiju transformācijai jēdzienu tīklos. Ontoloģijas kļuva plaši izplatītas, kad 2004. gadā tika izstrādāta tīmekļa ontoloģiju valoda OWL (Web Ontology Language) un virkne ontoloģiju veidošanas rīku. Ņemot vērā acīmredzamas līdzības starp ontoloģijām un jēdzienu tīkliem un valodā OWL izstrādāto pieejamo ontoloģiju skaitu, ir uzsākts pētījums par ontoloģiju izmantošanu jēdzienu tīklu ģenerēšanai. Šī pētījuma rezultātā ir plānots izstrādāt programmatūru OWL ontoloģiju transformācijai jēdzienu tīklos, lai atvieglotu pasniedzēju darbu, kas jēdzienu tīklus izmanto zināšanu vērtēšanā. Šajā rakstā ir aprakstīta pētījuma sākuma fāze, kurā tiek identificēti atbilstošie OWL ontoloģijas un jēdzienu tīkla elementi. Attēlojums starp OWL valodas konstrukcijām un jēdzienu tīkla elementiem ir aprakstīts šajā rakstā. Definētie attēlojumi ir pārbaudīti manuāli pielietojot tos eksistējošām ontoloģijām. Rakstā ir attēlots arī viens no iegūtajiem jēdzienu tīkliem. Atklātās attēlojumu nepilnības ir aprakstītas. Manuālā transformācija atklāja, ka ir problēmas ar jēdzienu tīklu elementu atspoguļošanu gadījumos, kad Būla operācijas tiek lietotas klases definēšanā. Līdzīgas problēmas rodas arī atspoguļojot īpašību mantošanu un hierarhiju.*

**Graudina V. OWL ontology transformation into concept map**

*This paper is dedicated to ontology transformation into concept maps. Since development of Web Ontology Language (OWL) in 2004 and many tools for ontology construction, ontologies have become quite widespread. Taking into account obvious similarities between ontologies and concept maps and number of available OWL ontologies, the research of ontology usage for concept map generation is undertaken. As the result of this research would be software for OWL ontology transformation into concept maps to reduce workload of teachers*

*who use concept maps for knowledge assessment. This paper describes initial phase of research, where corresponding elements of ontology and concept map have been identified. Mappings between OWL language constructs and elements of concept map are proposed in this paper. Defined mappings have been checked by applying them to existing ontologies manually. One of the manually created concept maps is shown in this paper. Incompleteness of proposed mappings discovered has also been described. Manual transformation showed that there were problems with representation of concept maps in case Boolean operations were used to define other classes. The same problems occurred while representing inheritance and hierarchy of properties.*

**Граудиня В. Трансформация OWL-онтологий в карты понятий**

*Данная работа посвящена теме трансформации онтологий в карты понятий. С момента развития Web Ontology Language (OWL) (Языка Сетевых Онтологий) в 2004 году и многих инструментов для создания онтологий, онтологии стали весьма распространенными. Учитывая явное сходство между онтологиями и картами понятий и число доступных OWL онтологий, проведено исследование использования онтологий в создании карт понятий. Результаты етого исследования будут использованны для создания программного обеспечения предназначенного для трансформации OWL онтологий в карты понятий с целью уменьшить нагруску препадавителей, которые используют карты понятий для проверки знаний студентов. Данная работа описывает начальную фазу исследования, в которой сопоставляются элементы онтологии и карт понятий. В данной работе представлено отображение конструкций языка OWL на элементы карты понятий. Получнные отображеные было проверено путем их применения к уже существующим онтологиям; также в работе описаны недостатки и недоработки предложенных отображение. Одна из полученных вручную карт понятий представлена в статье. Мануальная трансформация выявила проблемы связанные с представлением карт понятий в случаях, когда для определения других классов использовались Булевские операции. Данная проблема также проявлялась при представлении наследования и иерархии свойств.*