ELECTRONICS AND ELECTRICAL ENGINEERING
ISSN 1392 – 1215 2012. No. 4(120)
ELEKTRONIKA IR ELEKTROTECHNIKA

T 120 SYSTEM ENGINEERING, COMPUTER TECHNOLOGY
SISTEMŲ INŽINERIJA, KOMPIUTERINĖS TECHNOLOGIJOS

A Methodology for Engineering OWL 2 Ontologies in Practise Considering their Semantic Normalisation and Completeness

L. Nemuraite, B. Paradauskas
Department of Information Systems, Kaunas University of Technology,
Studentu str. 50, Kaunas, Lithuania, phone: +37037453445, e-mails: lina.nemuraite@ktu.lt. bronius.paradauskas@ktu.lt

crossref http://dx.doi.org/10.5755/j01.eee.120.4.1462

Introduction

Ontology related concepts and theories were investigated a long ago but only during the last years they were empowered with technologies as RDF, RDF(S), OWL, currently OWL 2 [1], and SPARQL capable to bring ontologies to the Semantic Web and various computerised environments. The vision of the Semantic Web addresses ontologies as main means of sharing common understanding and reasoning about relevant problem domains (e.g. information systems; medicine; software, electrical and other kinds of system engineering, analysis of social relations, etc.) among different communities of humans and machines.

In today practise, ontologies often are designed from scratch or reverse-engineered from existing data sources. During creation and maintenance of ontologies, developers usually are facing with problems of supporting evolution of ontologies and preserving integrity of ontology data. Though these objectives are very similar to those of databases and information systems (IS) [2−5], the purpose of ontologies is different, so the theory and best practises of databases cannot directly be applied to ontology design. In current IS methodologies, databases are designed on the base of concept models, which also have much in common with ontologies. The goal of the paper is to consolidate existing methods and best practises in ontologies, databases and conceptual modelling fields into one seamless methodology for practical developing ontologies.

The rest of the paper is structured as follows. Section "Related works" analyses criteria and problems in the area of developing ontologies. The following sections present the example of ontology and the methodology proposed illustrated with the example fragments. Finally, we summarize conclusions and envisage future works.

Related works

For being a sound knowledge base for reasoning in a problem domain, the ontology should satisfy certain quality criteria as correctness, consistency, extendibility, minimality, competence, completeness, and normalization [6−10]. These criteria were analysed by different authors long before modern semantic technologies have emerged e.g. [6, 7].

Different authors are focused on different aspects of ontology development. Methodology of Gruninger and Fox is based on competence and completeness of ontology [7]. Authors working in Description Logics are giving most attention to consistency. Rector emphasizes normalisation, modularity and reuse [8]. In practical ontologies as, e.g., Systematized Nomenclature of Medicine − Clinical Terms (SNOMED CT), the mentioned criteria are hardly satisfied. After significant auditing efforts to improve its consistency, correctness and completeness, SNOMED CT still contains semantically duplicate primitive and defined concepts, uses primitive concepts while these can be defined in terms of other primitives and roles, and has other redundancies of semantic content. Therefore, it cannot be efficiently maintained and used in semantic applications. For preventing such situations in creating the Common Ontology of Lithuanian Language and domain-specific ontologies for semantic search in SemantikaLT project [11], we present the methodology for developing ontologies suitable for semantic search related both with structured and unstructured information resources.

Proton ontology [12] is another example of a practical experience in developing ontologies. Proton was developed from KIM ontology with preconceived position of avoiding shortcomings characteristic to progressively gathered ontologies as SNOMED CT. Proton is divided into 4 modules of class hierarchies comprising so-called "schema ontologies" that approximately correspond to our notion of "base ontologies". Other classifications are treated as "topic hierarchies". Proton authors argue that hierarchies of schema ontologies and topics should not be mixed up; furthermore, they do not seek for comprehensive modelling of semantics of topics considering this semantics as very vague. In our approach, we also separate the base ontology (or concept schema) not only from its supplementing classification ontologies but also from inferable constructs; further, we present rules for building those classification ontologies, which could be defined, and combining different classification criteria.

Ontology example

For illustrating our methodology, we will take a very simplified ontology of a genealogy tree of persons (Fig.1) merged with a simplified ontology "friend of a friend" (in practise, both ontologies are kept as separate modules). Such ontology could be used for analysing human relations based both on kindred and friendship (and others). The base ontology contains only non-inferable constructs. For specifying instances, all mandatory properties, appearing in a base ontology, should be inserted into definitions of individuals. Genealogy tree ontology, supplemented with examples of inferable object properties, is presented in Fig. 1 as UML class diagram where UML classes represent OWL 2 classes, UML attributes – OWL 2 data properties, UML associations – OWL 2 object properties (association end names correspond to names of direct and inverse object properties; class, related to an association end, corresponds to a range of the property, and the class related to the opposite end – to a property domain). Inferable association ends are marked with "/" symbol. Really, much more object properties may be inferable (e.g. has_great_grandparent, has_great_grandchild, etc.).

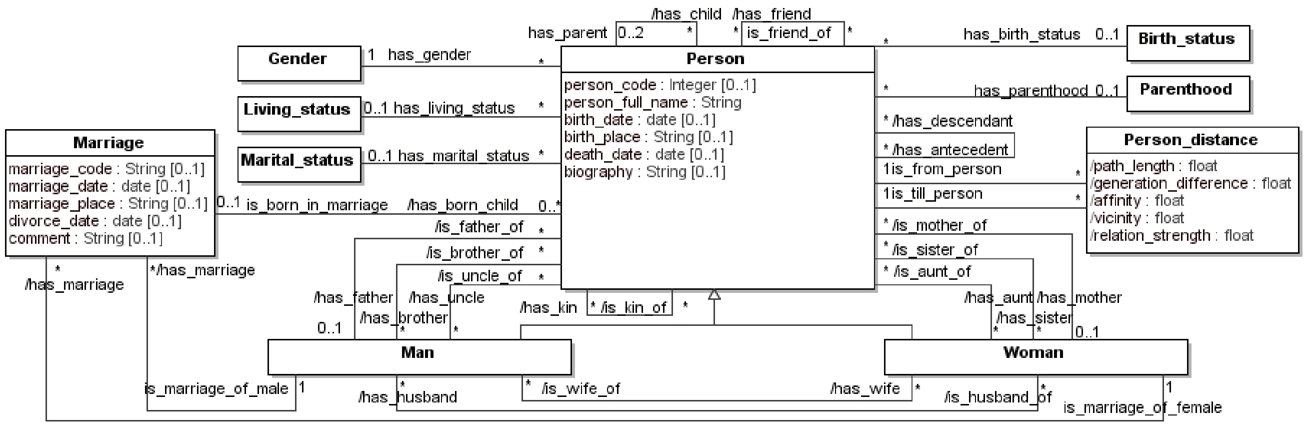

Fig. 1. Genealogy tree ontology with base classes and part of derivable constructs

The Kindred ontology (Fig. 2) allows classifying individuals according kindred, marital or birth status, parenthood and, possibly, a lot of categorization schemes that are not shown here.

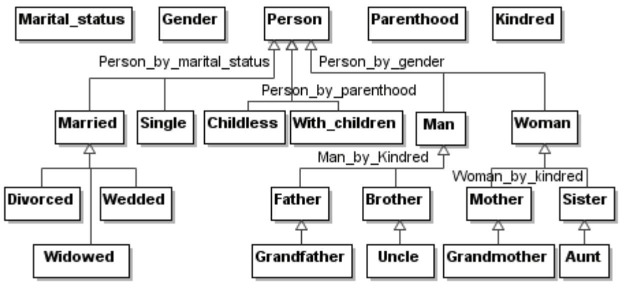

Fig. 2. Fragment of Kindred ontology

It is worth noting that each categorization scheme represented as UML generalization set should be equivalent to a primitive class (e.g. `Person` or `Marriage`) in a base ontology.

Methodology of ontology development

Our methodology consolidating different aspects concerned in considered works consists of 8 steps. It was proved using Protegé ontology editor, Pellet and Hermit reasoners, and ARQQuery engine for SPARQL.

1st step. Formulating ontology requirements. The goal of ontology is to give answers to competency questions about problem domain [7] so the initial requirements to ontology can be specified as informal competence questions and formalized into OWL Description Logics (DL) axioms what corresponds to requirement specification in Information System Development methodology. For satisfying criteria of competence, ontology under development should be able to answer to these questions irrespectively of any functionality that may be related with that domain. In contrast, conceptual models are designed for fulfilment of functional requirements, and there always exists mutual dependencies between structure and behaviour represented by conceptual models.

The competency questions should be defined in such a way that more complex questions depend on more simple ones. As requirements in IS, these questions are used to validate the ontology completeness. For example, ontology of Genealogy tree should give answers to the following questions: 1) What persons are fathers? 2) What persons are uncles? 3) What marriages are childless? 4) Who is father of person X? 5) Who are antecedents of person X? 6) Who are descendants of person X? 7) Who are kin of person X? 8) What is birth status of person X? 9) What is marital status of couple X? 10) What persons are friends of person X? 11) What persons are both kin and friends of person X? 12) What is distance between generations of person X and person Y? 13) What is strength of relation between person X and person Y? For being semantically complete, ontology also should differentiate between persons having father and having no father, kin and not kin, etc.

2nd step. Creating base ontology. Ontology classes can be divided into primitive classes (for which we are not able to give a complete axiomatic definition) and defined, or inferable, classes that can be derived from the primitive ones according to complete definition (consisting of necessary and sufficient conditions) [6]. Primitive classes should comprise well-defined subsumption hierarchy in which every class may be subsumed by at most one parent class; all sibling classes should be disjoint (but not necessarily covering their parent). Primitive values and value types also should comprise closed subsumption hierarchies where primitive values should be disjoint but primitive value types may overlap. Furthermore, every object or data property should have a domain and a range, and every universal restriction `allValuesFrom` should have `an` existential restriction `someValuesFrom` in the class or one of its superclasses; every individual should be an instance of exactly one most specific class in a primitive subsumption hierarchy. These requirements lead to semantically normalised ontologies that, similarly as in conceptual models and databases, meet requirements of explicitness and modularity of their components, allow their independent evolution and maintenance, and help to avoid update anomalies. For achieving minimality, the base ontology should include a minimal set of constructs allowing inferring the remaining ones.

3rd step. Creating instances. Individuals often are created as a part of ontology and reasoners are working on the whole ontology with its individuals. Concept models also can incorporate instances (differently from schema that does not include instances) but usually this may be done only for explanation or validation purposes. For applications, instances are stored in databases where integrity of instances is supported with native functionality of databases and additional components implementing integrity constraints that are directly not supported in Database Management Systems (DBMS).

Ontologies having many instances (millions and trillions of triples) use incremental reasoning techniques or combine them with processing ontology data in databases. For better maintainability, it is purposeful to separate instances from the base ontology and store them separately. In particularly, for maintaining integrity of ontology the best solution is to store instances in a relational database using method like described in [13, 14] as the native functionality of ontology tools does not apparently support integrity of instances. It means that ontology editors conforming to Description Logics do not prevent users from making incomplete (implicit) descriptions of individuals. The second important requirement for creating instances is preventing of inserting inferable constructs. Therefore, inferable constructs should be described in separate ontology using base ontology import.

4th step. Creating supplementing ontologies. The purpose of inferable classes is to support answering of competence questions. In contrast, inferable classes do not comprise part of conceptual models; they are rather comparable to views in a database that emerge in the stage of the detailed design.

In ontology, inferable classes may comprise multiple subsumption hierarchies that correspond to multiple contexts of reasoning in ontology. Similarly as in conceptual models, such hierarchies may be sub-divided into separate categorization schemas where each schema corresponds to a single context (Fig. 2). Different categorization schemas are represented as different hierarchies of defined classes; top parents of these hierarchies are mutually equivalent to each other and to a corresponding primitive class. For multiple categorizations, such structures may become cumbersome.

In practise, inferable classes comprise a volatile, fast changing part of ontology, which may have unlimited number of emergent contexts. For clarity and maintainability, it is worth to separate these different contexts from each other and from the base ontology, and to merge them in various ways for analysing different contexts or their compositions. For example, when analysing persons who are both kin and friends we should merge the Genealogy tree, Kindred and Friendship ontologies, along with individuals of the Genealogy tree.

5th step. Formalizing competence questions. 1−3 competence questions are easy answered by reasoning on supplementing ontologies having proper axioms, e.g.

```
Father ≡ Person ⊓∃ has_child ⊓has_gender.male.(1)
```

Other kinds of questions (e.g. 4−11) cannot be resolved by classification. These questions may be answered using rules represented in Semantic Web Rule Language (SWRL), e.g. SWRL rules are able inferring father (2), descendants and antecedents (3, 4), all kin of each person (5), kin who also are friends (6), etc.:

```
Person(?x) ⊓has_parent(?x, ?y) ⊓
   has_gender(?y, male) ⇒ has_father(?x, ?y),     (2)

has_child(?x, ?y) ⇒ has_descendant(?x, ?y)
has_child(?x, ?y) ⊓has_descendant(?y, ?z) ⇒
  has_descendant(?x, ?z),                          (3)

has_parent(?x, ?y) ⇒ has_antecedent(?x, ?y)
has_antecedent(?z, ?x) ⊓has_parent(?x, ?y) ⇒
  has_antecedent(?z, ?y),                          (4)

has_antecedent(?x, ?y) ⇒ has_kin(?x, ?y)
has_descendant(?x, ?y) ⇒ has_kin(?x, ?y)
   has_antecedent(?x, ?y) ⊓has_descendant(?y, ?z),
 DifferentFrom (?x, ?z) ⇒ has_kin(?x, ?z),        (5)

has_kin(?x, ?y) ⊓is_friend_of(?x, ?y)  ⊓
  DifferentFrom (?x, ?y) ⇒
    is_kin_and_friend_of(?x, ?y).                  (6)
```

After the inference, kin of the particular person may be obtained via a SPARQL query (here `gen` is a prefixed name of ontology `Genealogy_tree`)

```
select distinct ?kin {gen:Vita_Binkiene
  gen:has_kin ?kin}.                               (7)
```

Competence questions 12 and 13 are not so easy to define even with SPARQL 1.1; they require writing additional code and were specially introduced for illustrating practical needs of manipulating with ontologies. Though we cannot formalize 12−13 questions with ontology axioms or SWRL rules, we must introduce additional classes (as `Person_distance` in Fig. 1) and properties into ontology for ensuring answering these questions by using other means.

For satisfying competence requirements, ontology should give reasonable answers not only to all competence questions but also to counter questions for ensuring completeness of the classification. Description logics allows defining a limited variety of possible questions that could be answered by classifying individuals according certain criteria (e.g. marital status) into corresponding subsumption hierarchies. As a rule, the classification criteria should be explicit (8), in particular cases expressed via a dedicated object property (e.g. `marital_status`), because it would be impossible inferring a complement of a set (e.g. single persons having no individuals related via the object property "`has_marriage`") (9) though it would be easy finding individuals having that value (persons married in some marriage) (10):

$$\texttt{Single} \equiv \texttt{Person} \sqcap \texttt{marital\_status.single}, \tag{8}$$

$$\texttt{Single} \equiv \texttt{Person} \setminus \texttt{Married}, \tag{9}$$

$$\texttt{Married} \equiv \texttt{Person} \sqcap \exists\ \texttt{has\_marriage.Marriage}. \tag{10}$$

Properties dedicated for explicit specification of classifying values cause redundancies in ontology. For ensuring consistency of the ontology, we should define additional axioms, e.g., we should state that persons having even been married should have marital status "wedded", or "divorced", or "widowed", and persons by marital status could be categorized only as `Married` or `Single`:

$$\texttt{Person\_by\_marital\_status} \equiv \texttt{Married} \sqcup \texttt{Single}, \tag{11}$$

$$\begin{aligned}\texttt{Married} \equiv\ & \texttt{Person\_by\_marital\_status} \sqcap \\ & (\texttt{has\_marital\_status.wedded} \\ & \sqcup \texttt{has\_marital\_status.divorced} \\ & \sqcup \texttt{has\_marital\_status.widowed}) \\ & \sqcap \exists\ \texttt{has\_marriage.Marriage}.\end{aligned} \tag{12}$$

The additional axioms are trade-off w.r.t. explicitness, consistency and minimality of ontology. It is possible avoiding redundancies by using SPARQL query language (with its extension SPARQL 1.1), which has the greatest possibilities for discovering knowledge from ontologies as it is capable for finding complement of a set without explicitly stating, e.g., that person was not married in marriage; answering recursive queries, computing values etc. In such a case, minimality wins but explicitness is lost

```
select ?single {?single rdf:type gen:Person.
 NOT EXISTS {?single has_marriage ?marriage}}.(13)
```

6th step. Ensuring completeness. Completeness of ontology is directed toward ensuring its competence and differs from notion of completeness in data concept models where the later is understood as semantic completeness, which covers structural completeness, instance completeness, and beyond [3]. In short, semantic completeness for ontologies is satisfied via normalizing ontology and validating ontology model. During maintenance stage, instance completeness for a whole ontology may not be always achieved, but, according to our methodology, it is always partially achieved for created subset of individuals.

Grüninger and Fox [7] propose to achieve ontology completeness by proving completeness theorems w.r.t. the competency questions. The point of these theorems is that 1) ontology and its individuals should semantically entail competency questions; 2) ontology and its individuals should be consistent with competency questions; 3) ontology and its individuals should semantically entail counter questions w.r.t. competency ones; 4) all models of ontology should agree on its extensions. In our methodology, all of these requirements are taken into account; however, they are dependent on developer and there are no automatic means for ensuring them.

7th step. Validating ontology. There are several aspects of ontology validation. The consistency of ontology is analysed by a reasoner according Description Logics rules. Ontology is consistent if it has no contradictory statements. However, it does not mean that, for example, if a class `Marriage` has object property axiom "`is_marriage_of_female exactly one Woman`, and the individual `Marriage1` has object property assertions "`is_marriage_of_female Saule Gudiene`" and "`is_marriage_of_female Saulyte`", such ontology would be recognized as inconsistent by reasoner. It would be a case only if it would be stated that `Saule Gudiene` and `Saulyte` are disjoint. Otherwise, because of the open world assumption, reasoner assumes that `Saule Gudiene` and `Saulyte` is the same person. Therefore, every assumption about the problem domain under consideration should be made explicit.

The normalization of ontology is not analysed by ontology editors. Normalized ontology can be created using Formal Concept Analysis (FCA) [15]. In practise, FCA is rather used for validating ontologies created by human after analysing the problem domain. FCA allows multiple parents therefore it can be applied for ontologies describing multiple contexts related with so called conceptual scaling [15]. The example of multi context lattice for a part of ontology in Fig. 2 is presented in Fig. 3.

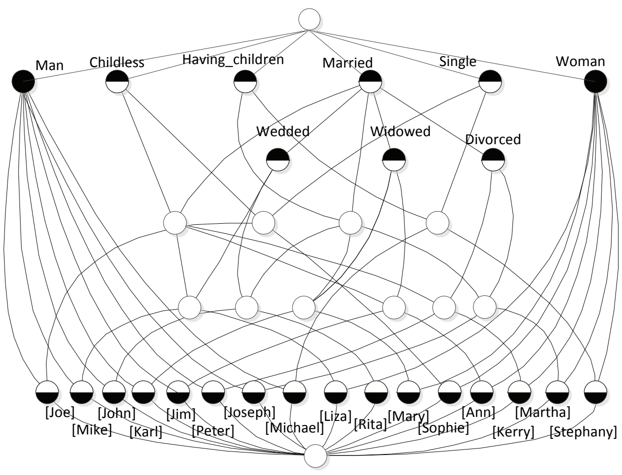

Fig. 3. Concept lattice for part of ontology in Fig. 2

For using FCA for validating a base ontology or a single categorization schema, FCA requirements should be restricted to a single subsumption hierarchy (the concept lattice for classifying persons w.r.t. marital status, including "Marital_status_unknown" for explicitness of conceptualization, is presented in Fig. 4).

For validating consistency and normalisation of ontology, the sufficient amount of individuals should be described. It does not mean that thousands of individuals are needed for validation. Conversely, we need one complete instance of ontology (i.e. ontology model) that encompasses at least one instance for each construct of ontology.

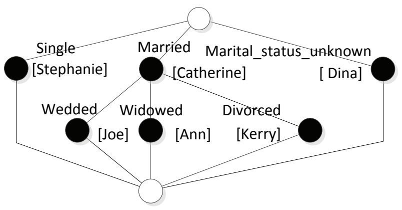

Fig. 4. Concept lattice for single subsumption hierarchy

Formal context involves a set of instances that may be represented in a table [15]. Such set may be taken as an initial approximation of ontology model; however, it is not enough. Here we can recall the universal instance assumption (UIA) from the theory of relational databases [2]. UIA requires that the relations in a database all be projections of some common instance of a domain under consideration. That means a database represents a single relation over a universal scheme – a join of all relations in the database. The universal relation scheme assumption (URSA) requires that an attribute represents a single role of a class of entities i.e. it has the same meaning everywhere in the considered database context (e.g. "date" meaning "`birth_date`" or "`death_date`" should be defined by different names). UIA and URSA are requirements of the universal intension and universal extension of the context. By comparing the UIA and URSA to the definition of formal context in FCA [15], we can conclude that formal context is not sufficient for counting number of individuals, comprising a full instance of a base ontology as it does not take into account concept relations (object properties). We propose to use for this the extended concept structure (Fig. 5) where functional dependencies (i.e. functional or inverse functional object properties) between concepts are taken into account.

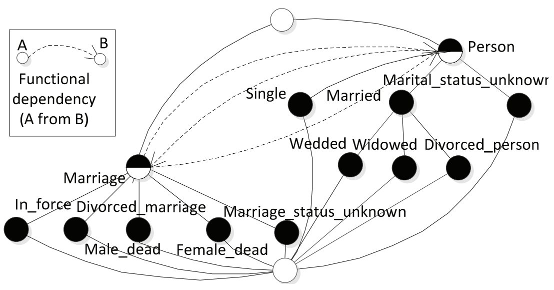

Fig. 5. Extended concept structure for estimating number of individuals for ontology with concept lattice presented in Fig. 4

For estimating a number of instances, we use the following rules. One object is required for representing each most specific concept in `SubClassOf` hierarchy if all siblings in that hierarchy are disjoint and covering their parents; open disjoint hierarchies require one object for each concept. Bijective functional dependencies between two concepts require one object for each concept; injective functional dependencies (i.e. `FunctionalObject Properties`) – two objects for a domain concept and one object for a range to analyse cases when the relation exists and when it does not.

For representing reflexive relations between objects, we need two objects for representing one concept from the corresponding formal context, and three objects for transitive relations; constraints on a number of objects participating in relations also should be taken into account. One value or value object is required for each value from closed disjoint value sets. The number of required objects may be reduced as the same object can (or even must) participate in several relations if this participation is not forbidden by axioms and does not cause inconsistency of ontology.

Validation of supplementing ontologies requires additional individuals related with conceptual scaling comprising a many-valued context. For each of multiple contexts, we should add an individual for each value of an attribute, which acts as a categorization type for a chosen conceptual scale.

For example, an instance of ontology, which concept lattice for persons categorized according marital status is presented in Fig. 4, should involve 5 individuals of different types of `Marriage` (i.e. "`In_force`", "`Divorced_marriage`", etc); each marriage requires for male and female. It means, a minimal complete instance should include 12 individuals of type "`Person`" (6 men and 6 women) and 5 individuals of "`Marriage`" while formal context in Fig. 4 requires just 6 individuals of type "`Person`".

8th step. Validating individuals in the maintenance stage. Ontology representation should be designed so that it would be possible to extend it monotonically, i.e. without changing the existing definitions [6]. For introducing new categorization schemes, new properties may be needed to add to the base ontology without which values for counter questions may be not inferable, as it was shown with `marital_status` example. These new properties should not violate aforementioned quality criteria of ontology. Under such requirements, it is purposeful to fill ontologies with data by the means of DBMS using our proposed method [13, 14] or similar as ontology development tools are not able to constrain creation of individuals for ensuring integrity of ontology.

Conclusions and future works

1. The investigation of practical needs for developing ontologies for the Semantic Web or Enterprise applications has shown that for discovering meaningful information and effective maintenance of ontologies it is desirable to keep them consistent, normalized and precise as much as possible. In particularly, for employing the power of reasoners, all properties that could not be derived should be explicitly specified. It requires hard efforts; however, the usage of reasoners allows formulating simple queries that could be directly represented in natural language – such intention is currently pursued by the Semantic Web as well as other computer-supported environments of human activities.

2. Our proposed methodology, accumulated from existing methods and practical experiences, helps developing ontologies satisfying quality criteria encompassing correctness, consistency, extendibility, minimality, competence, completeness, and normalization. Under such requirements, it is purposeful to store and maintain ontologies by the means of DBMS as ontology development tools are not able to constrain creation of individuals for ensuring integrity of ontology.
3. For validating ontologies, we propose to ensure completeness of ontology model by taking into account functional object properties and other dependencies for evaluating a number of individuals required for creating complete ontology models.
4. Our future work is related with investigation of applying the proposed methodology for developing practical ontologies, in particularly, related with SemantikaLT project, and, in general case, when ontology data are obtained from various sources and may be imprecise, incomplete, and ambiguous. In our methodology, we have embraced the capability to work with incomplete and ambiguous ontology models keeping them consistent; however, the feedback from practical applications is needed for assuring avoidance of further undesirable issues.

References

1. **Motik B., Patel–Schneider P. F., Parsia B.** OWL 2 Web Ontology Language Structural Specification and Functional–Style Syntax. – W3C Proposed Recommendation. – W3C, 2009. – 134 p.
2. **Maier D.** The theory of relational databases. – London: Pitman, 1983. – 438 p.
3. **Paradauskas B., Nemuraitė L.** Duomenų bazės ir semantiniai modeliai. – Monografija. – Kaunas: Technologija, 2002. – 264 p.
4. **Nenortaitė J., Butleris R.** Improving Business Rules Management through the Application of Adaptive Business Intelligence Technique // Information Technology And Control. – Kaunas, Technologija, 2009. – Vol. 38. – No. 1. – P. 21–28.
5. **Kalnius R., Eidukas D.** Probability Model Quality of Informations Systems // Electronics and Electrical Engineering. – Kaunas: Technologija, 2009. – No. 4(92). – P. 13–18.
6. **Gruber T. R**. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. – Technical report, KSL–93–04. – Knowledge Systems Laboratory, Stanford University, 1993. – 23 p.
7. **Grüninger M.**, **Fox M. S**. Methodology for the Design and Evaluation of Ontologies // Proceedings of the IJCAI Workshop on Basic Ontological Issues in Knowledge Sharing. – Menlo Park Calif.: AAAI Press, 1995. – P. 1–10.
8. **Rector A. L.** Normalisation of ontology implementations: Towards modularity, re–use, and maintainability // Proceedings Workshop on Ontologies for Multiagent Systems (OMAS) in conjunction with European Knowledge Acquisition Workshop, 2002. – P. 1–16.
9. **Cordi V., Mascardi V**. Checking the completeness of ontologies: a case study from the semantic web // Proceedings of the Italian Conference on Computational Logic (CILC'2004). – Quaderno del Dipartimento di Matematica, University of Parma, 2004. – Vol. 390. – P. 1–15.
10. **Cornet R., Abu–Hanna A**. Two DL–based methods for auditing medical terminological systems // AMIA Annu Symp Proc., 2005. – P. 166–170.
11. „SemantikaLT" − Lietuvių kalbos sintaksinės–semantinės analizės ir paieškos sistema lietuviškam internetui, tekstynams ir viešojo sektoriaus taikymams (2012–2014). – VP2–3.1–IVPK–12–K–1–07. – 2012. – 230 p.
12. **Terziev I., Kiryakov A., Manov** D. Base–upper–level–ontology Guidance. – EU–IST Project IST–2003–506826 SEKT Deliverable D1.8.1, 2005. – 70 p.
13. **Vyšniauskas E., Nemuraitė L., Paradauskas, B.** Hybrid method for storing and querying ontologies in databases // Electronics and Electrical Engineering. – Kaunas: Technologija, 2011. – No. 9(115). – P. 67–72.
14. **Vyšniauskas E., Nemuraitė L., Butleris R., Paradauskas, B**. Reversible lossless transformation from OWL 2 ontologies into relational databases // Information technology and control. – Kaunas : Technologija, 2011. – Vol. 40. – No. 4. – P. 293–306. DOI: 10.5755/j01.itc.40.4.979.
15. **Wille R.** Formal Concept Analysis as Mathematical Theory of Concepts and Concept Hierarchies // Formal Concept Analysis. – Berlin Heidelberg: Springer–Verlag, 2005. – Vol. 3626. – P. 1–33.

Received 2011 12 13
Accepted after revision 2012 01 14

L. Nemuraite, B. Paradauskas. A Methodology for Engineering OWL 2 Ontologies in Practise Considering their Semantic Normalisation and Completeness // Electronics and Electrical Engineering. – Kaunas: Technologija, 2012. – No. 4(120). – P. 89–94.

The goal of the paper is to present a practical methodology for developing OWL 2 ontologies on the base of existing theories and best practices for ontologies and concept models. In practise, ontologies often are developed from scratch because formal theories are too far from practical needs, or they are focused on one of quality aspects. The presented methodology allows developing ontologies with regards to quality criteria, applied in various ontology development methodologies, and is supplemented with rules for estimating a number of individuals required for ontology validation. Ill. 5, bibl. 15 (in English; abstracts in English and Lithuanian).

L. Nemuraitė, B. Paradauskas. Praktinė OWL 2 ontologijų inžinerijos metodika, nagrinėjanti jų semantinį norminimą ir išsamumą // Elektronika ir elektrotechnika. – Kaunas: Technologija, 2012. – Nr. 4(120). – P. 89–94.

Straipsnyje pateikiama praktinė OWL 2 ontologijų kūrimo metodika, grindžiama esamomis konceptų modelių ir ontologijų kūrimo teorijomis ir praktine patirtimi. Praktikoje ontologijos dažnai kuriamos nepaisant teorijų, kurios sunkiai praktiškai pritaikomos ar sukoncentruotos į vieną iš ontologijų kokybės aspektų. Straipsnyje pateikiama metodika atsižvelgia į daugelį kriterijų, taikomų įvairiais metodais kuriant ontologijas; be to, ji apima taisykles, leidžiančias nustatyti individų skaičių, reikalingą ontologijos kokybei patikrinti. Il. 5, bibl. 15 (anglų kalba; santraukos anglų ir lietuvių k.).