Pak. j. life soc. sci. (2008), 6(1): 50-58

Pakistan Journal of Life and Social Sciences

Ontology Development Methodology for Semantic Web Systems

Amjad Farooq and Abad Shah
Computer Science and Engineering Department
University of Engineering and Technology, Lahore-Pakistan

Abstract

Ontology is considered as backbone of every semantic web system. It formally describes data and descriptive knowledge of a domain in machine understandable form for automatic processing and interpreting. Some methodologies have been proposed for developing ontology for semantic web systems. Most of them are based on the 'built and fixed' approach. In that, first initial version of ontology is built and improved iteratively until domain requirements are satisfied. In this way the basic principles of software engineering are not followed properly. These methodologies mainly focus on data rather than the descriptive knowledge of domain. These methodologies mainly work on specification and implementation phases and design phase lacks in proper attention. Moreover the design and implementation phases of theses methodologies are difficult to identify and separate. In this paper we present a methodology for ontology development, which may overcome these weaknesses

Keywords: Ontology, development methodology, ontology engineering, Semantic Web

Introduction

Relevant information, retrieval and processing from current web have become very critical issue nowadays because most of its contents are not machine understandable. The knowledge of most domains exists in descriptive form and not in machine understandable form. This issue is being addressed through Semantic Web (Lee *et al.*, 2001). World Wide Web Consortium (W3C) has recommended that the descriptive knowledge of a domain along with its data needs to be expressed in some logic-based languages to enable it machine-processable. That formal description of data and descriptive knowledge is usually termed as ontology.

Corresponding author: Amjad Farooq
Computer Science and Engineering Department,
University of Engineering and Technology, Lahore-Pakistan
Email: amjadfarooq@uet.edu.pk

Ontology is coded in logic-based language and is like any other software. Thus developing it according to software engineering standards can results in high quality ontology at low development and maintenance cost.

Several ontology development methodologies have been proposed by AI community (Asunción and David, 2000; Paul *at el*., 2005). Most of these methodologies are based on natural language processing (NLP) and machine learning techniques. Orientation of these methodologies is web-agents view-point rather than web-contents formalization. The work on ontology development was boosted when the idea of semantic web was envisioned. Different methodologies of developing ontology are reported in literature (Uschold and King, 1995; Fernandez *at el.*, 1997; Menzel at el, 1994; Clyde, 2002) and substantial work is still needed in this area. Although, these methodologies distribute the work in logical phases i.e. specification and formalization but the design phase is not explicitly focused. Since the design phase is the most important stage of software development and is considered as a backbone of software development (Baresi and Morasca, 2007; Richard and André, 2007), therefore there should be an explicit design phase in web-ontology development process.

In this paper, we outline a methodology for ontology development for semantic web. The key features of this methodology are: (i) follow-up the software engineering standards, (ii) guideline of different activities in algorithmic form in order to automate them (iii) simple and practical-oriented strategy for development activities. Proposed methodology consists of three phases: (a) specification, (b) design and (c) formalization. Testing and documentation activities are carried out during and final stage of each phase.

Remaining part of this paper is organized as follows. A brief overview of semantic web and existing ontology development methodologies is given in Section 2. Our proposed ontology development methodology is given in Section 3. A case study on proposed methodology is presented in Section 4 via a case study. The paper concludes with recommendation for future work in Section 5.

Ontology Development for Semantic Web

Semantic web has various advantages over non-semantic web (i.e. current web). A comparative study is given below.

Semantic web encompasses actual content along with their formal semantics. Here formal semantics are machine understandable content, generated in logic-based languages such as Web Ontology Language (OWL) recommended by W3C. Through formal semantics of content, computers can make inferences about the data i.e. to understand data resource along with its relations to other data. Current web does not have formal semantics of its contents. These contents are machine-readable but not machine understandable.

Current web is like a book, having multiple hyperlinked documents. In book scenario, index of keywords are there but the context in which those keywords are used is missing. There are no formal semantics of keywords in indexes. To check which one is relevant, we have to read the corresponding pages of that book. Same is the case with current web. In semantic web, this limitation is eliminated via ontologies, where data is given well-defined formal meanings, understandable by machines.

Through literature survey, it has been determined that inaccessible part of the web is about five hundred times more than what search engines find (Siegfried *at el.*, 2003). It is estimated that there are billion pages of information available on the web, and only a few of them can be reached via traditional search engines. In semantic web, formal semantics of data are available via ontologies and completely accessible to semantic search engines.

Semantic web is the web of ontologies having data with formal meanings. This is in contrast to current web which contains virtually boundless information in the form of documents. The semantic web, on the other hand, is about having data as well as documents that machines can process, transform, assemble, and even act on it in useful ways.

There are various web resources that may be very useful in our everyday activities. It is difficult to locate them in current web because they are not annotated properly by the metadata. In semantic web there can be a network of related resources which are easy to locate and use.

Semantic web has many other advantages in terms of information searching, accessing, extracting, interpreting and processing. Moreover, semantic web can have inference or reasoning capability and have lower communication cost.

There are a growing number of methodologies for ontology development (Cristani and Cuel, 2005). Mostly these methodologies focus on specification and formalization of ontology and do not concentrate on its design phase. In KBSI IDEF5 methodology (Menzel *at el*., 1994), data about domain is collected and analyzed. Then 'built and fixed' strategy is used to create ontology. Ushold and King (Uschold And King, 1995) proposed an ontology development methodology. In this methodology, after identifying purpose of ontology, it is captured and then coded. In MethOntology (Fernandez *at el*., 1997), after preparing ontology specification, knowledge is acquired and analyzed to determine domain terms such as concepts, relations and properties and then formalization is started. After that, evaluation and documentation is performed. In (Clyde, 2002)], a methodology based on collaborative approach has been proposed. In its first phase, design criteria for the ontology, specified boundary conditions for the ontology and set of standards for evaluating ontology, are defined. In second phase, the initial version of ontology is produced, and then through iterative process the desired ontology is obtained. Software 'Built and Fixed' approach is followed, which leads to heavy development and maintenance cost. In (Helena and João, 2004), following steps are proposed for ontology constructing: specification, conceptualization, formalization, implementation and maintenance. Knowledge acquisition, evaluation and documentation are performed during each phase. There are some other approaches investigating the transformation of a relational model into an ontological model. Ontology is built, based on the database schemas. These approaches mainly use reverse engineering theory. In (Stojanovic, 2002) ontology is constructed from conceptual database schemas using a mapping process. To carry out the process, it is necessary to know the underlying logical database model that will be used as source data. Information from a relational schema is captured, analyzed and transformed into ontology and then it is refined and evaluated. In (Fernandez and Gomez-Perez, 1997), initial version of ontology is constructed from database schemas and then it is refined using a collection of queries that are of interest to the database users. In (Irina and Bela, 2005), ontology is constructed from HTML forms used to communicate with database.

Proposed Methodology

The proposed ontology development methodology mainly consists of three phases (i) Specification (ii) Design (iii) Formalization. In specification phase domain vocabulary is defined; resources and their interaction within domain are identified and constrained. In the design phase an ontology model is built and then formalized in third phase, so-called formalization phase. All these phases are briefly described below:

3.1 Specification Phase

This phase consists of seven activities: (a) Domain vocabulary declaration, (b) Identifying resources and assigning them to proper groups, (c) Identifying Axioms, (d) Identifying relationships and assigning them proper names, (e) Identifying data-characteristics and assigning them proper names, (f) Applying constraints and (g) Validation.

Brief descriptions of all these activities are given below:

Domain Vocabulary Declaration: Domain vocabulary is the foundation of a web-ontology. It is prepared with consensus and consultation of domain experts, ontology engineers and web engineers involved in development of semantic web application, in order to avoid semantic heterogeneity. Commonly occurring words and phrases denoting domain concepts are properly named. Attributes and verbs along with their description are also included in domain vocabulary list. All vocabulary terms are defined in a namespace, referenced by some Universal Resource Identifier (URI). (b) Identifying resources and assigning them to proper groups: In web context, a resource is any thing that has URI or can be referenced by an URI. Therefore if some concept has number of instances then all those instances are grouped in a class and that class is included in the resource-list. Multiple classes can belong to same page, and each page is represented by a URI, therefore that page is also included in the resource-list. Similar pages may be grouped into a page-class, represented by an URI; therefore that page-class should be included in the resource-list. (c) Identifying Axioms: The structural knowledge about, how the resources interact with each other, may be specified in terms of axioms. These are sentences written by using domain vocabulary. These represent declarative knowledge about concepts, and these are always accepted to be true without any proof. Axioms also represent semantics about behavior and properties of concepts. Normally they are interpreted as rules for concepts and we can drive different information about concepts from those axioms. All domain axioms are listed in this activity. (d) Identifying relationships and assigning them proper names: From axioms listed in previous step, interactions between resources are determined and proper names are assigned to each relationship from domain vocabulary produced in step a). If some one is missing, go step (a) and define them. Inverse name is also defined and listed for each relationship to make application more knowledge-enriched. (e) Identifying data-characteristics and assigning them, proper names: A characteristic is a specific feature, attribute, or element used to describe a resource. Each characteristic has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other characteristics. Assign proper name to each data-characteristic from domain vocabulary. If some one is missing, go step (a) and define them. Inverse name is also defined and listed for each name to make application more knowledge- enriched. (f) Applying constraints: A domain for a named-relationship specifies which resources are potential subjects of statements, having that named-relationship as predicate. Here the statement is the basic element of preliminary web-ontology model, it has triplet format and the predicate is one of them (i.e. subject, predicate and object). Domain of named-relationship consists of classes because classes encompass resources. There can be multiple classes in the domain of named-relationship. A range for a named-relationship specifies which resources may become objects of statements those have that named-relation as predicate. Again, there can be multiple classes in the range of a named-relationship. (g) Verification: Although this activity should be carried out in parallel to each of the above steps, but after completion of preliminary web-ontology model, it should be tested again for its consistency, correctness and completeness with the help of domain experts. For consistency the defects such as, using more than one name for same resource, and an individual assignment to two mutually disjoint classes, are diagnosed. In completeness testing, the defects such as omission of domain resources and the omission of relationship are diagnosed, and similarly in correctness testing the use of incorrect relationship, aggregation & specialization of classes and cardinality of relationship are tested. Semantic heterogeneities are also determined and dissolved in this step. The constraints on domain and range values of each object property and datatype property are also verified in the testing activity.

```
1. Define all concepts found in target domain. Each concept is
   properly named using domain vocabulary. Each concept corresponds to
   a class.
2. Add each class in a list - Resource.
3. for i = 1 to n (where n is the total resources found in the domain)
        IF Resource[i] is a domain (i.e. subdomain) THEN
              goto step 1 (recursive call)
        end-if
4. List all schema axioms of target domain
5. N = size of Resource array
6. For i = 1  to N
              1. Find Resource [i]'s relationships to other
                 resources from axioms. Assign proper name to each
                 relationship
              2. Add  each named-relationship of  Resource[i] in a
                 set OP[i]
              3. Add its data-characteristics(s) in a set DP[i]

7. namedRelationshipSet  = OP[1] union OP[2] - - - union OP[N]
8. datatCharacteristicSet = OP[1] union OP[2] - - -union OP[N]
9. For each element of namedRelationshipSet
                 1. list its domain classes
                 2. list its range classes
                  [Organize then in tabular form]
10. For each element of datatCharacteristicSet
          i.   list its domain classes
         ii.   specify datatype for its value or specify its literal
        iii.   specify its integrity constraints (if any)
```

Figure 1: Ontology Conceptualization in Pseudo-form

Design Phase

Design phase is considered as backbone of software. It enables software logic technology-independent, the requirements are further refined, reduce formalization (coding) efforts, make software more manageable, adaptable, and maintainable. The processing of design phase mainly uses the report generated by specification phase, and transforms it into some algorithmic or pseudo form so that it can be coded easily in any computer language in order to make it executable. Since ontology (schema and document) is based on Resource Description Framework model, therefore we design a model so-called RDF model, from preliminary ontology model generated in previous phase. The design phase is represented in algorithmic form in Fig. 2. RDF Model: This model consists of triples. A triple contains three components: (i) subject, (ii) predicate and (iii) object. Each name in RDF model is a URI reference or a literal. All these names are unambiguously defined in respective namespaces. Each resource and its instance are represented using a set of statements describing the same resource. Their identities are given via URIs and those identifiers are globally unique.

(i) Predicates: All characteristics of resources and their relationships as determined in previous phase are taken as predicates. A predicate can have multiple subjects or objects.

(ii) Subjects: All domain classes of characteristics and relationships of resources as determined in section (f) of previous phase, are taken as resources. Subject can be a blank node. A blank node is not a URI reference or a literal. It is simply a unique node that can be used in one or more RDF statements.

(iii) Objects: All range classes of relationships as determined in section (f) of previous phase and literals (values of characteristics) are taken as objects. An object can also be a blank node if there are multiples object-values. A literal represents values

such as numbers and dates by means of a lexical representation. A literal can also be represented by a URI.

Formalization Phase

It is said that writing code is not a problem, understanding problem and designing a solution for that problem, is a problem. Several tools are available for the assistance of ontology creation, such as Protégé-2000 (Natalya at el., 2001). This is very simple tool, graphically create semantic web contents in OWL and/or RDFS, and we can create code very quickly and easily. There are online validation services of W3C for checking ontology syntactical validity and consistency. Similarly for testing the knowledge-richness and reasoning capabilities of ontology, reasoners are there such as (Evren and Bijan, 2004).

Case Study

In this section, we present a case study to illustrate proposed methodology. For this purpose, we develop an ontology for CS&E department of university. In this paper we have illustrated the "RAM- Research Activity Management" section of university domain, omitting other sections due to space limitation. RAM manages data about research groups, research areas, researchers, research papers and about the management of different research groups.

(i) Specification Phase

a) Domain Vocabulary: There may be a number of research groups, each has a URI, and so these are grouped in a class so-called "*ResearchGroup*". A person entitled as Director directs each research grouped. There may be several researchers, having unique URIs; we have grouped them in a class called "*Researcher*". Each researcher may have different data-characteristics such as name, job-title, joining-date, email and mobile number; these data-characteristics are formally termed as *hasName, hasJobTitle, hasStartingDate, hasEmail,* and *hasCell* respectively. Research paper is a very common concept used in research activities. There may be several papers written by different authors, as each paper may be referenced by a URI, therefore we group them in a *ResearchPaper* class. Each research paper has title, author(s), abstract and publishing-year, paper-type and body section. These characteristics are formally termed as *hasAuthor, hasTitle, hasAbstract, hasPublishingYear, hasCategory,* and *hasTextURI* respectively. Each group conducts its research into different research areas, where each area has certain title along with short description and is unique-identifier. Description and unique-identifier characteristics are formally termed as *hasDescription* and *hasId* respectively where the title is already termed as hasTitle. All instances of research area are grouped in a class, formally named as *ResearchArea.* A research paper has a category that depends on the medium where it is published. It may be published in local conference, in an international conference, in research journal or may be a chapter in a book. It has been recommended that a separate class should be used for holding paper category information, and we have named that class as *paperCategory*. Each instance of paperCategory may have id, title and short description.

```
1.  Default Namespace selection
2.  Ontology namespace with prefix identifier
3.  OWL namespace  xmlns:owl =http://www.w3.org/2002/07/owl#
4.  RDF namespace xmlns:rdf ="http://www.w3.org/1999/02/22-rdf-syntax-
    ns#"
5.  RDFS namespace  xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
6.  XML Schema datatype namespace
       xmlns:xsd ="http://www.w3.org/2001/XMLSchema#">
7.  xml:base insertion
          (i.e xml:base="http://www.owl-ontologies.com/uet-named.owl">)
8.  search existing suitable ontologies for reuse
        IF they are found THEN add them in import list ENDIF
9.  Apply integrity constraint on datatype properties
    (value-constraints, cardinality constraints, functional constraints)
10. For each namedRelationship in namedRelationshipSet
        a.  create triples (predicate[subject, object])
        b.  assign namedRelationship  to predicate
        c.  assign domainClassName to subject and
        d.  corresponding rangeClassName to object
11. For each dataCharacteristic in datatCharacteristicSet
        a.  create triples (predicate[subject, object])
        b.  dataCharacteristic  to predicate
        c.  assign domainClassName to subject and
        d.  corresponding datatype  to object
12. Create RDF graph/model through integration of output of
    steps (10) and (11)
```

Figure 2: Ontology design in pseudo-form

b*) Resources identification and grouping in term of classes: RearchGroup*: There may be number of research groups, each has a URI, so these are grouped in a class so-called "ResearchGroup". *Researcher:* There may be several researchers, having unique URIs; we have grouped them in a class called

"Researcher". *ResearchPaper:* Paper is a very common concept used in research activities. There may be several papers written by different authors, as each paper may be referenced by a URI, therefore we group them in a *ResearchPaper* class. *ReseachArea:* Each group conducts its research into different research areas, where each area has certain title along with short description and is uniquely identified by a URI; we group them in a *ResearchArea* class. *PaperCategory***:** a research paper may be a published in local conference, in an international conference, in research journal or a chapter in a book. Each individual of paperCategory may have id, title and short description.

c*) Axioms about Resources of Domain: A ResearchGroup has ResearchArea; A ResearchGroup has a director; A ResearchArea has deputyDirector; A director and deputyDirector are Researchers; A Researcher may be a student, faculty or a software engineer; A researcher writes ResearchPaper; A researchPaper may be a National Conference Paper; A researchPaper may be an International Conference Paper; A researchPaper may be a Journal article; A researchPaper may be chapter in some Book; A researchPaper has author(s); An author is a Researcher; A researchPaper has a Title; A researchPaper has text; A researchPaper has publishing year.*

d) Relationship: Identifying and Naming: Relationship between ResearchGroup class and Researcher class is named as hasDirector. *hasDeputyDirecto*r is a named-relationship between ResearchArea class and Researcher class. *hasAutho*r is a named-relationship between researchPaper class and Researcher. *hasArea is a* named-relationship that exists between ResearchArea class and ResearchGroup class. Relationship between ResearchPaper class and PaperCategory class is named as hasCategory. Similarly relationship between Researcher class and ResearchArea classes is named as *hasResearchArea.* We have chosen a few relationships, by excluding others due to space limitation.

e) Data-characteristics of Resources

Name	**Domain Class**	**Range Class**
HasId	*ResearchGroup* *ResearchArea* *Researcher* *PaperCategory*	Number datatype
HasTtile	*ResearchGroup* *ResearchArea* *Researcher* *ResearchPaper* *PaperCategory*	String
HasEmail	*Researcher*	String
HasName	*Researcher*	String
HasCell	*Researcher*	Number
HasAffiliation	*Researcher*	String
HasStartingDate	*ResearchGroup* *ResearchArea* *Researcher*	Date
HasText	*ResearchPaper*	PageURI
HasPublishingYear	*ResearchPaper*	Number

Table 4: Data-elements with domain and range constraints

f) Domain and range specification of named-relations (Constraints on named-relations)

Name	**Domain Class**	**Range Class**
HasDirector	*ResearchGroup*	*Researcher*
HasDeputyDirector	*ResearchArea*	*Researcher*
Has Area	*ResearchGroup*	*ResearchArea*
Has Author	*ResearchPaper*	*Researcher*
HasCategory	*ResearchPaper*	*PaperCategory*
HasResearchArea	*Researcher*	*ResearchArea*

Table 5: Resource relationships along with domain and range constraints

g) Validating

With the help of domain experts we evaluated outputs of all activities of this phase to check their completeness and correctness. Different types of tests as mentioned in validation section of specification phase in proposed section are performed before switching to next phase.

(ii) Design Phase: The output of specification phase is transformed into a RDF model which encompasses triples. We have represented the output of previous phase in triples using the format; *Predicate [Subject, Object]*

is_a [Author, Class]
is_a [Person, Class]
subClassOf [Author, Person]
is_a [hasCategory, ObjectProperty]
hasRange [hasCategory, PaperCategory]
hasDomain [hasCategory, ResearchPaper]
is_a [belongToPage, ObjectProperty]
is_a [hasArea, ObjectProperty]
is_a [hasAuthor, ObjectProperty]
hasDomain [belongToPage, HardCodedContent]
hasDomain [Author, ResearchPaper]
hasRange [belongToPage, PageClass]
is_a [Director, Class]
is_a [Faculty, Class]
------
subClassOf [Faculty, Person]
is_a [hasTitle, DatatypeProperty]
hasDatatype [hasTitle, String]
hasDirector [ResearchGroup, Director]
hasArea [ResearchGroup, ResearchArea]
hasAuthor [ResearchPaper, Author]
hasCategory [ResearchPaper, PaperCategory]
hasResearchArea [Researcher, ResearchArea]
is_a [hasStartingDate, DatatypeProperty]
hasDatatype [hasStartingDate, DATE]

Figure 3: Sample slice of triples

(iii) Formalization

We have used Protégé Version 3.2.1 ontology editor for formalization of RDF model produced in previous step.

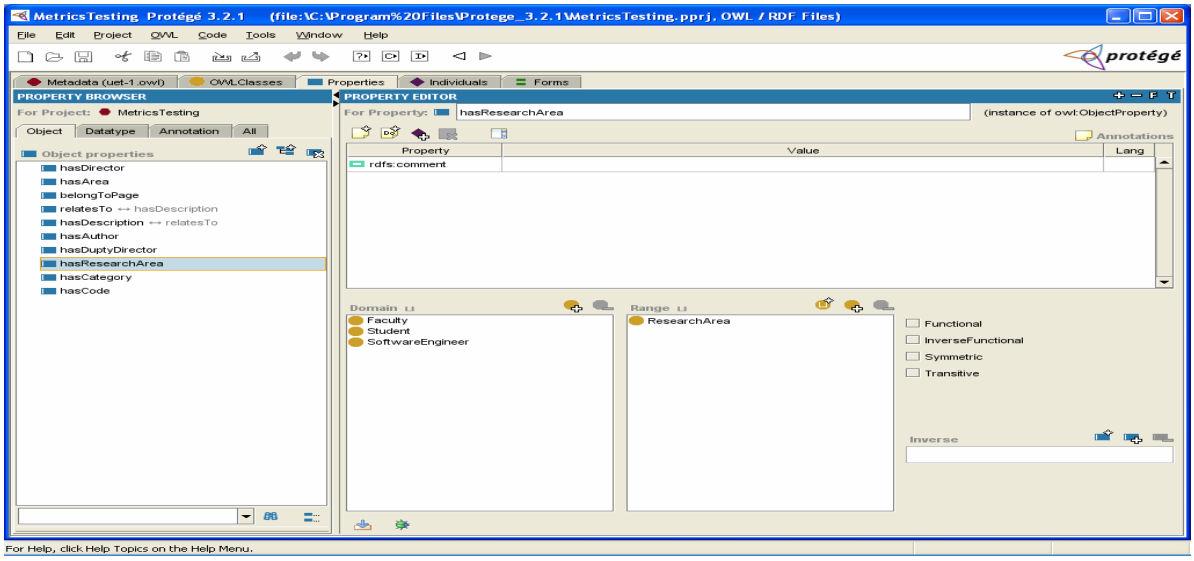

Figure 4: Protégé user-interface

```
<rdf:RDF xmlns="http://www.owl-
ontologies.com/uet-1.owl#"
    xml:base="http://www.owl-ontologies.com/uet-
1.owl"
xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
xmlns:rdfs="http://www.w3.org/2000/01/rdf-
schema#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
syntax-ns#"
xmlns:owl="http://www.w3.org/2002/07/owl#">
   <owl:Ontology rdf:about="">
      <owl:versionInfo rdf:datatype="&xsd;string"
        >Web-ontology for Research Actvity
Management Domain</owl:versionInfo>
      <rdfs:comment
rdf:datatype="&xsd;string"></rdfs:comment>
   </owl:Ontology>
   <owl:Class rdf:ID="Author"/>
   <owl:Class rdf:ID="Director"/>
   <owl:Class rdf:ID="DuptyDirector"/>
   <owl:Class rdf:ID="Faculty">
      <rdfs:subClassOf rdf:resource="#Person"/>
   </owl:Class>
   <owl:DatatypeProperty rdf:ID="hasAffiliation">
      <rdfs:domain rdf:resource="#Researcher"/>
      <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>
   <owl:ObjectProperty rdf:ID="hasArea">
      <rdfs:domain rdf:resource="#ResearchGroup"/>
      <rdfs:range rdf:resource="#ResearchArea"/>
   </owl:ObjectProperty>
   <owl:ObjectProperty rdf:ID="hasAuthor">
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
      <rdfs:range rdf:resource="&xsd;string"/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID="hasId">
      <rdfs:domain>
         <owl:Class>
            <owl:unionOf
rdf:parseType="Collection">
               <owl:Class rdf:about="#Author"/>
               <owl:Class rdf:about="#Director"/>
               <owl:Class
rdf:about="#DuptyDirector"/>
               <owl:Class
rdf:about="#PaperCategory"/>
               <owl:Class rdf:about="#Person"/>
               <owl:Class
rdf:about="#ResearchArea"/>
               <owl:Class rdf:about="#Researcher"/>
               <owl:Class
rdf:about="#ResearchPaper"/>
            </owl:unionOf>
         </owl:Class>
      </rdfs:domain>
      <rdfs:range rdf:resource="&xsd;int"/>
   </owl:DatatypeProperty>
   <owl:DatatypeProperty rdf:ID="hasName">
      <rdfs:domain>
         <owl:Class>
            <owl:unionOf
rdf:parseType="Collection">
```

Figure 5: Sample slice of ontology

Conclusion and Future Work

We have outlined a methodology for developing ontology for semantic web. The key consideration of our methodology is that ontology should be developed according to software engineering principles, in order to achieve high quality ontology with low maintenance cost. Guidelines for conceptualization and design phases are provided in algorithmic forms. Now we are working on this methodology, to (semi-) automate these phases via developing a software tool, based on these algorithm.

References

Asunción Gómez-Pérez, David Manzano-Macho. "OntoWeb D.1.5 A survey of ontology learning methods and techniques", IST Project IST-2000-29243

Baresi, L. and Morasca, S. Three empirical studies on estimating the design effort of web applications. ACM Trans. Software. Eng. Methodol. 16, 4, Article 15. 2007.

Clyde W. Holsapple "A collaborative approach to ontology design" Communications of the ACM . ISSN:0001-0782 . 2002

Cristani, M; Cuel, R.: A Survey on Ontology Creation Methodologies. International Journal on Semantic Web and Information Systems, Vol. 1, No. 2. 2005

Evren Sirin, Bijan Parsia: Pellet: An OWL DL Reasoner. Proceedings of the 2004 International Workshop on Description Logics (DL2004), Whistler, British Columbia, Canada, June 6-8, 2004.

Frasincar, F., Houben, G. Hypermedia presentation adaptation on the semantic web. In Adaptive Hypermedia and Adaptive Web-Based Systems, Second International Conference, AH 2002, volume 2347 of Lecture Notes in Computer Science. Springer, ISBN 3-540- 43737-1. 133-142. 2002.

Fernandez, M., Gomez-Perez, A. And Juristo, N. METHONTOLOGY: From Ontological Art Towards Ontological Engineering, AAAI-97 Spring Symposium on Ontological Engineering, Stanford University, March 24-26th. 1997.

Helena Sofia Pinto, João P. Martins. "Ontologies: How can They be Built?" Knowledge and Information Systems , 6: 441–464. 2004.

Irina Astrova and Bela Stantic. Reverse Engineering of Relational Databases to Ontologies: An Appoach Based on an Analysis of HTML Forms. 2004.

Lee, T., J. Hendler, and O. Lassila. The semantic web. Scientific American. 2001.

Jones D. "Methodologies For Ontology Development" Department of Computer Science, University of Liverpool, Liverpool, U.K., L69 7ZF.

KBSI. The IDEF5 Ontology Description Capture Method Overview, KBSI Report, Texas. 1994.

K.-I. Lin and H. Chen. Automatic information discovery from the invisible web. In Proceedings of the The International Conference on Information Technology: Coding and Computing (ITCC'02), pages332–337, 2002.

Natalya F. Noy, Michael Sintek, Stefan Decker, Monica Crubézy, Ray W. Fergerson, and Mark Creating SemanticWeb Contents with Protégé-2000. A. Musen, Stanford University. 2000.

Natalya F. Noy and Deborah L. "Ontology Development 101: A Guide to Creating Your First Ontology" Stanford University, Stanford, CA, 94305

Paul Buitelaar, Philipp Cimiano, Bernardo Magnini (eds.) Ontology Learning from Text: Methods, Evaluation and Applications Frontiers in Artificial Intelligence and Applications Series, Vol. 123, IOS Press. 2005.

Reif, G., Gall, H., Jazayeri, M. WEESA - Web Engineering for Semantic Web Applications. Proceeding of 14th International World Wide Web Conference, Chiba, Japan. 2005.

Richard N. Taylor, André van der Hoek, "Software Design and Architecture The once and future focus of software engineering," fose, pp. 226-243, Future of Software Engineering (FOSE '07), 2007

Stojanovic, L.; Stojanovic, N.; Volz R. Migrating data-intensive Web Sites into the Semantic Web. Proceedings of the 17th ACM symposium on applied computing (SAC), ACM Press, 2002, pp. 1100-1107. 2002.

Sherman C. and G. Price. The Invisible Web: Uncovering Information Sources Search Engines Can't See. CyberAge Books. 2001.

Siegfried Handschuh, Raphael Volz and Steffen Staab "Annotation for the Deep Web". IEEE Intelligent Systems Published by the IEEE Computer Society. 2003.

Uschold, M. And King, M. Towards A Methodology for Building Ontologies. IJCAI-95 Workshop on Basic Ontological Issues in Knowledge Sharing, Montreal, Canada. 1995.

Uschold, M. And Gr Ninger, M. (1996) "Ontologies: Principles, Methods and Applications", Knowledge Engineering Review, 11(2), 93-137.

Updegrove A. , The Semantic Web: An interview with the Tim Berners-Lee. 2005.