A Semantic Web Framework for Teaching Logic Circuits

Francisco-Edgar Castillo-Barrera
Engineering Faculty
Universidad Autónoma de San Luis Potosí
Dr. Manuel Nava 8, Zona Universitaria poniente
78290 San Luis Potosí, México
ecastillo@uaslp.mx

Carolina Medina-Ramírez
Department of Electrical Engineering
Universidad Autónoma Metropolitana, México
Av San Rafael Atlixco 186, Col.Vicentina
09340 Distrito Federal, México
cmed@xanum.uam.mx

Jose Emilio Labra Gayo
Department of Computer Science
Universidad de Oviedo
C/Valdes Salas s/n 33007-Oviedo, España
jelabra@gmail.com

S. Masoud Sadjadi
School of Computing and Information Sciences
Florida International University (FIU)
11200 SW 8th St, Miami, USA
sadjadi@cs.fiu.edu

Abstract

Ontology-based systems have been used to facilitate teaching and learning. Ontologies have proven to be a very useful artifacts to represent a domain and as an important component in specific applications for giving semantics. In logic circuits domain, ontologies have been employed for teaching logic gates (xor, or, not, nand), and this approach has been deemed as an effective way for capturing and using the knowledge of the logic gates on assembling circuit systems. This knowledge can be reused by new students gaining time and reducing circuits manufacturing costs. In addittion, the correct assembling among logic gates and the right output of a circuit can be validated by using semantic techniques. In this paper, we describe a semantic web technique based on a core ontology, a reasoner and SPARQL queries for teaching and learning circuits based on logic gates. We use an example and a prototype to explain our approach.

1. Introduction

New scenarios for self-learning of logic circuits are neccesary during the design phase. These scenearios have to ensure the funcionality expected by the students and simulating the behavior of their circuits. These scenarios help them to prevent economic losses. Another important factor to consider is the circuit models reusability [19]. The time for developing a complex circuit using a semantic circuit repository decrease the cost of the project and reduce the learning curve of new students in the project. In this context, semantic technologies seem relevant. We can use Ontologies[32] in order to represent a circuit based on logic gates (and, or, not, etc.) and to verify the circuit design. Each connection of the circuit can be validated by means of ontology properties [32] and reasoners [27]. The new knowledge obtained for each part of the circuit assembled, can be stored in an ontology [21] written in OWL-DL [39] by means of metadata (is stored as an XML file), in this way the knowledge is capitalized [30][23]. This knowledge can be used by new developers or new members of the project to reduce manufacture time. In consequence, the company decreases costs. The circuits behavoir can be modelled by SPARQL queries. In fact, a complex circuit could be represented by one SPARQL query.

The rest of the paper is structured as follows. In Section 2 we give the related work for teaching logic circuits based on ontologies and SPARQL queries. In Section 3 we briefly explain concepts about Semantic Web Techniques, Ontologies, Core Ontologies, Reasoners and SPARQL queries. Section 4 we describe our approach for teaching of logic circuits in a Semantic Web Framework. In Section 5 we show the feasibility of our technique by describing an example and a prototype called Itzamna. Finally, in Section 6 we conclude our work.

2. Related work

The ontologies based on logic gates for teaching is mostly represented by work of Robal et al.[29] who wrote an ontology-based intelligent learning object for teaching

the basics of digital logic. Robal's ontology is oriented for teaching the basics of digital logic, our ontology is made for validating the right connections among logic gates, for verifying the right output of the logic circuit built, and Students can create new circuits reusing the ontology. Another work by Sosnovsky and Gavrilova [33] was made by teaching and learning C programming based on the designed ontology.

3. Semantic Web Techniques

The Semantic Web [35][3][24] is the Tim Berners-Lee vision for representing information in the World Wide Web. He defines it as a web of data that can be processed directly and indirectly by machines [35]. This is a collection of standards, a set of tools [7], and a community that shares data. Semantic Technology is a concept in computer science which goal is to give semantics to data[10]. Supported by semantic tools [31] that provides semantic information about the meaning of words (RDF, SPARQL, OWL, and SKOS). *Semantic Web Techniques are methods and techniques based on semantic tools which allow us to manipulate information too*.

3.1 Ontologies

Ontologies are the key for Semantic Web goals. An Ontology [11][13][35][31][32] is defined by Gruber as *a specification of a conceptualization* [11] which defines the terms used to describe and represent a domain of knowledge, also is the model (set of concepts) for the meaning of those terms, thus defines the vocabulary and the meaning of that vocabulary, are used by students and applications that need to share domain information. More specifically, an ontology is a formal representation of knowledge with semantic content which allows us to obtain information. Such information can be retrieved by performing SPARQL queries [28] or using a rule-based inference engine [34]. In our case, the logic circuits is the domain of knowledge.

3.1.1 Core Ontologies

In philosophy, a **Core Ontology** [6] is a basic and minimal ontology consisting only of the minimal concepts required to understand the other concepts. It must be based on a core glossary that humans can understand. A **Core Ontology** is a complete and extensible ontology that expresses the basic concepts in a certain domain of knowledge. In this work we have built a core ontology which consists of a logic gates glossary which students of circuits understand well. We consider that these kind of ontologies support the reuse. Building these kind of ontologies do not require a complex methodology [9] to follow it, in fact, following the Ontology Development 101: A Guide to Creating Your First Ontology [8] or An eXtreme method for developing lightweight ontologies [16] are enough.

3.2 SPARQL Query Language

SPARQL is a query language for the Resource Description Framework (RDF) [22] which is a W3C Recommendation [38]. RDF Schema (RDFS) is extending RDF vocabulary for describing taxonomies of classes and properties. We use Web Ontology Language OWL [39] which extends RDF and RDFS. Its primary aim is to bring the expressive and reasoning power of description logic to the semantic web. In our learning scenario, Querying language is necessary to retrieve information [17] and verifying the correct output of the circuits. At this moment, we only have decided to explore semantic queries in SPARQL instead of applying another action such as: production rules [34].

3.3 Reasoners

A reasoner [27] is a program which its main task is checking the ontology consistency. It verifies if the ontology contains contradictory facts, axioms or wrong properties among concepts. Besides, new knowledge can be inferred after applying it. The most popular reasoners are Cerebra [25], FACT++ [37], KAON2 [26], Pellet [27], Racer [14], Ontobroker [5], OWLIM [20]. Pellet is an open-source Java based OWL-DL reasoner. In our verification process we use Pellet for checking the consistency of the logic circuit ontology and classify the taxonomy. We select the Pellet reasoner, because it gives an explanation when an inconsistency was detected.

4 Teaching Scenario in a Semantic Web Framework

Itzamna[1] is a factory framework of circuit models based on semantic techniques which focuses on maximising the level of reuse using logic circuits. One of the most important features of this framework is enabling knowledge reuse in logic circuits modelling using Semantic web techniques [4].

The aim of this framework is to allow to learn about logic circuits using a friendly interface and a graphical desing. Our main contributions are: we define a framework that allows us to reuse logic circuit for building new circuits. Second, our approach supports the validation of the output values obtained from the logic circuit during the design phase and Finally, our framework supports the learning and teaching of circuits based on logic gates. A prototype of the

[1] Itzamna is the name of an upper god of wisdom in Yucatec Maya mythology

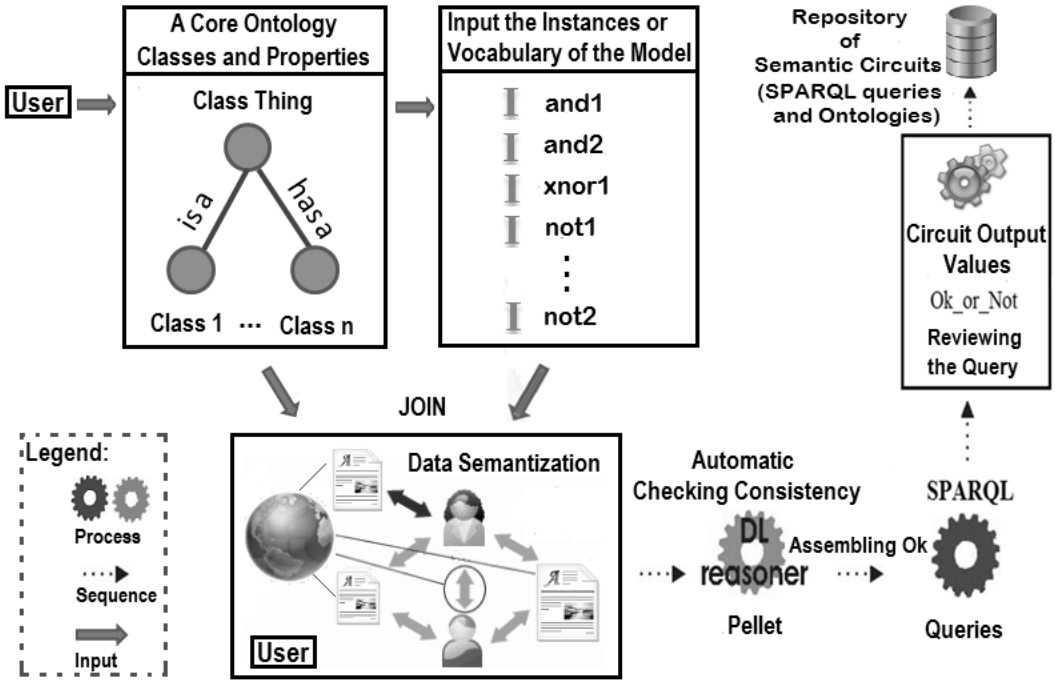

Figure 1. Semantic Web Techniques for Teaching Logic Circuits

framework involves a visual editor. The tool makes use of the library Flamingo and the Ribbon component [18] implemented in Java. We have used Jena API [11] and Java language [36] for programming that and NetBeans IDE 7.0 [2]. Each logic gate is represented in a graphic way and it can be assembled with another, see Figure 2. This is the first task to do by the students during his learnig process. Second, they have to introduce the information about the name and bit values of the circuits in the ontology. We have called this **Data Semantization process**, see Figure 1. This information also can be introduced by means of a text file (the *option Create Instances Vocabulary*). Itzamna transforms the user vocabulary (logic gates that the user needs for building his circuit) from a text file into an ontology instances. The third step will be to check the ontology consistency part of the Semantic Verification, see Figure 3. Semantic verification is the process which uses a core Ontology and Semantic Technologies (SPARQL queries) to guarantee the correct construction of logic circuits with specific connections and outputs. The semantics of assembling the logic gates are described with object properties. An important aspect of the logic gates to consider during the assembling is the Input and Output connections. A logic gate has one output, but different number of input connections. The logic gate connections are based on the output of one of them using as input in the others.

4.1 A Core Ontology for Logic Circuits

We propose a core ontology called **OntoCoreCircuit** which has the minimum concepts (logic gates) neccesary to represent the 1-bit Comparator circuit. And, Or, Xor, Not, Nand, Nor and Xnor are universal gates and they do not require to be validate by experts. Besides, we only need 3 or 5 competency questions to validate the ontology [12]. A Logic Gates Ontology was created for capturing and verifying information about the new logic circuit model during the graphical design. OntoCoreCircuit Ontology is built by means of classes using **n3** notation and relations among concepts. This is used by the ontology, because is a valid RDFS and OWL-DL notation. The Ontology use RDFS and OWL-DL language [39][15][40]. They are fundamentally based on descriptive logic languages. This Ontology consists of 3 Classes (*Circuit*,*Gate* and *Bits*), 35 Instances (*:and*, *:or*, *:not*, etc.), 10 Object properties (*:isTypeGate*, *:andOutput*, *:hasInput1*, etc.) and 1 Datatype property (*:hasName*).

5 Building a 1-bit Comparator in Itzamna Framework

A 1-bit comparator is a hardware electronic device that receives two bits (A and B) as input and determines whether one bit is **greater than**, **less than** and **equal** to the other bit.

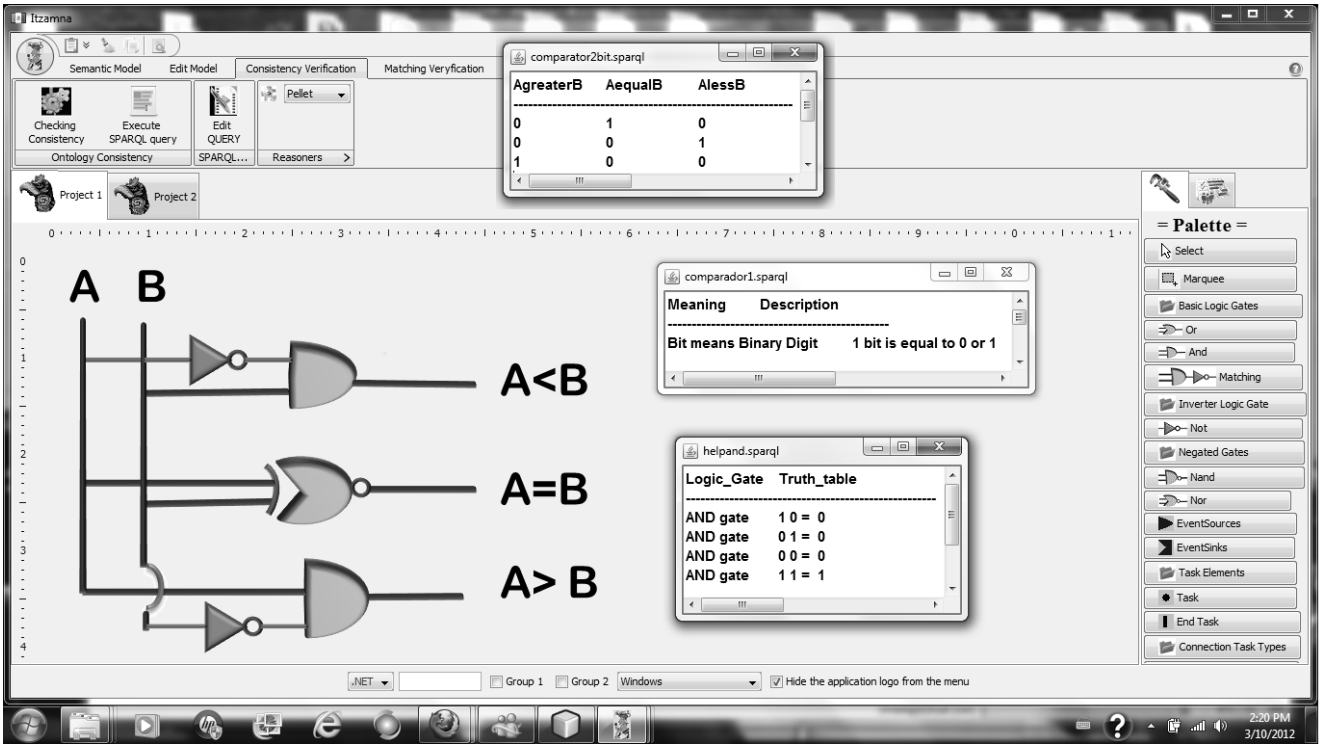

Figure 2. 1-bit Comparator circuit diagram and SPARQL queries

Table 1. 1-Bit Comparator Truth Table based on Instances

A	B	AB	A<B	A = B	A>B
:0	:0	:0_0	:0	:1	:0
:0	:1	:0_1	:1	:0	:0
:1	:0	:1_0	:0	:0	:1
:1	:1	:1_1	:0	:1	:0

This circuit has 2 bit binary inputs (A, B) and three single bit binary outputs (A $>$ B, A=B, A $<$ B). This kind of circuit can be extended for **2,3,...,n-bits** comparator circuit. For that reason, we have chosen this circuit for learning. The truth table using the instance notation is showed in Table 1. This circuit is built with 5 logic gates (2 not, 2 and, 1 xnor), as showed in Figure 2. The logic circuit model used for describe a 1-bit Comparator circuit was made in Itzamna Framework using its graphical interface of logic gates (the text in the image was adding for clarifying the circuit information), and is shown in figure 2. The input model (logic gates) is created by the user who selects classes and relation among concepts and he creates the logic gates instances (*:and1*, *:and2*, *:xnor1*, *:not1* and *:not2*). In this case the input model only has 5 logic gates and we can create its instances and relations among them using the Itzamna's menus (create instances vocabulary).

5.1 Assembling Verfication using The Pellet Reasoner

The Core Ontology written in OWL-DL, allow us to define restrictions which Pellet can verify during the consistency checking process. This action can be performed by menues in Itzamna framework, see Figure 3. For instance, the following code establishes that the *and* gate has only 1 output, because a FunctionalProperty is defined for *:andOutput* Object Property.

```
:xnorOutput a owl:ObjectProperty ;
    rdfs:domain   :Gate ;
    rdfs:range    :Bits ;
    rdf:type  owl:FunctionalProperty .
```

An interesting property of the ontology used in this work is a blank node. It is a node in an RDF graph representing a resource without URI or literal. We used it as variable. If we put the same blank node, the result for this node has to be the same. In our example below, _:c1, _:c2 and _:c3

are blank nodes (working as variables). The example shows how to *:xor1* and *:and2* gates are forced to have the same input (_:c2).

```
    # :xnor1 is a member of xnor gates
:xnor1 :isTypeGate  _:c1 .
    # :xnor1 requires 2 input values
_:c1   :hasInput2   _:c2 .
    # :not1 is a member of not gates
:not1  :isTypeGate  _:c3 .
    # :not1 requires only 1 input value
_:c3   :hasInput1   _:c2 .
```

A difference with Logic Programming Paradigm, we can check our types using ontologies. In particular when we create a new logic gate, for example *:and2*, we do not have to introduce all input and output values. In this case, it is only neccesary to establish the property relation *:and2* **:isTypeGate** *:and* . Besides, the ontology allow us to see circuits and gates saving in the ontology at the same time because the *Gate* class is a subclass of *Circuit*.

```
:Circuit         a owl:Class .
:Gate rdfs:subClassOf :Circuit .

:isTypeGate a owl:ObjectProperty ;
            rdfs:domain   :Gate ;
            rdfs:range    :Gate .
```

The *disjointWith* property allow to verify restrictions in the input model. For example a logic gate is not a bit, these two classes are different. Defining disjoint classes is also possible [1].

```
:Gate rdfs:subClassOf   :Thing ;
      owl:disjointWith :Bits .
```

All instances created, properties (object and datatype) established among instances, and blank nodes in the Ontology are checked by the reasoner Pellet during the consistency verification process.

5.2 Output Validation using a SPARQL Query

The last step after the reasoner have checked the ontology circuit consistency is to apply a SPARQL query for validating the correct output of 1-bit comparator circuit. In our case, we have defined a query which describes the circuit and obtain the output for given input values. We can think that SPARQL is the version of SQL for ontologies. Besides, we can use variables in the queries, constraints, filtering information, logic operators, if statements and more. Each triples (each line after) are linking by variables which begin with a question mark. In this code *?type1* and *?AB* are examples of variables. The same name of variable imply the same value to look for in the query. We can execute and edit queries in Itzamna framework because the Jena API allowed us to use SPARQL queries in our framework programmed in Java language. The following example shows the SPARQL query used in this work for validating the output values in our 1-bit Comparator circuit.

```
PREFIX   : <http://www.ejemplo.org/#>
PREFIX fn:
<http://www.w3.org/2005/xpath-functions#>

SELECT DISTINCT
       ?AgreaterB ?AequalB ?AlessB
WHERE
{
    :xnor1  :isTypeGate   ?type1 .
    ?type1 :hasName       ?xnorName .
    ?type1 :hasInput2     ?AB .
    ?AB      :xnorOutput   ?AequalB .

    BIND( if(?AB = :0_0, :0_1,
          if(?AB = :0_1, :0_0,
          if(?AB = :1_0, :1_1 ,:1_0)))
          AS ?ABneg )
    BIND( if(?AB = :0_0, :1_0,
          if(?AB = :0_1, :1_1,
          if(?AB = :1_0, :0_0 ,:0_1)))
          AS ?AnegB )

    :and1    :isTypeGate   ?type2 .
    ?type2  :hasName       ?and1Name .
    ?ABneg :andOutput   ?AgreaterB .

     :and2    :isTypeGate   ?type3 .
    ?type3  :hasName       ?and2Name .
    ?AnegB :andOutput   ?AlessB .
}
```

If the user wants to give an specific input values, only needs to change the variables *?AB* and *?Cin* for instances of the *Bits* class. For example: *:0_1 :xorOutput ?XorOutput* .

An optional step, when the logic circuit has been verified and validated, consists on storing the project independent of the ontology or include it in the core ontology. It is important to note that these challenges increase the reuse of this ontology and decrease the time in the development of future circuits. Benefiting the economy of companies (Knowledge Capitalization [30][23]). In our example, the code included in the core ontology was showed in Figure 3.

6. Conclusions

Teaching Logic Circuits by means of Semantic Web Techniques is possible with core ontologies, reasoners, and SPARQL queries. Ontologies are usually expressed in a logic-based language (Description-Logic), enabling detailed, sound, meaningful distinctions to be made among

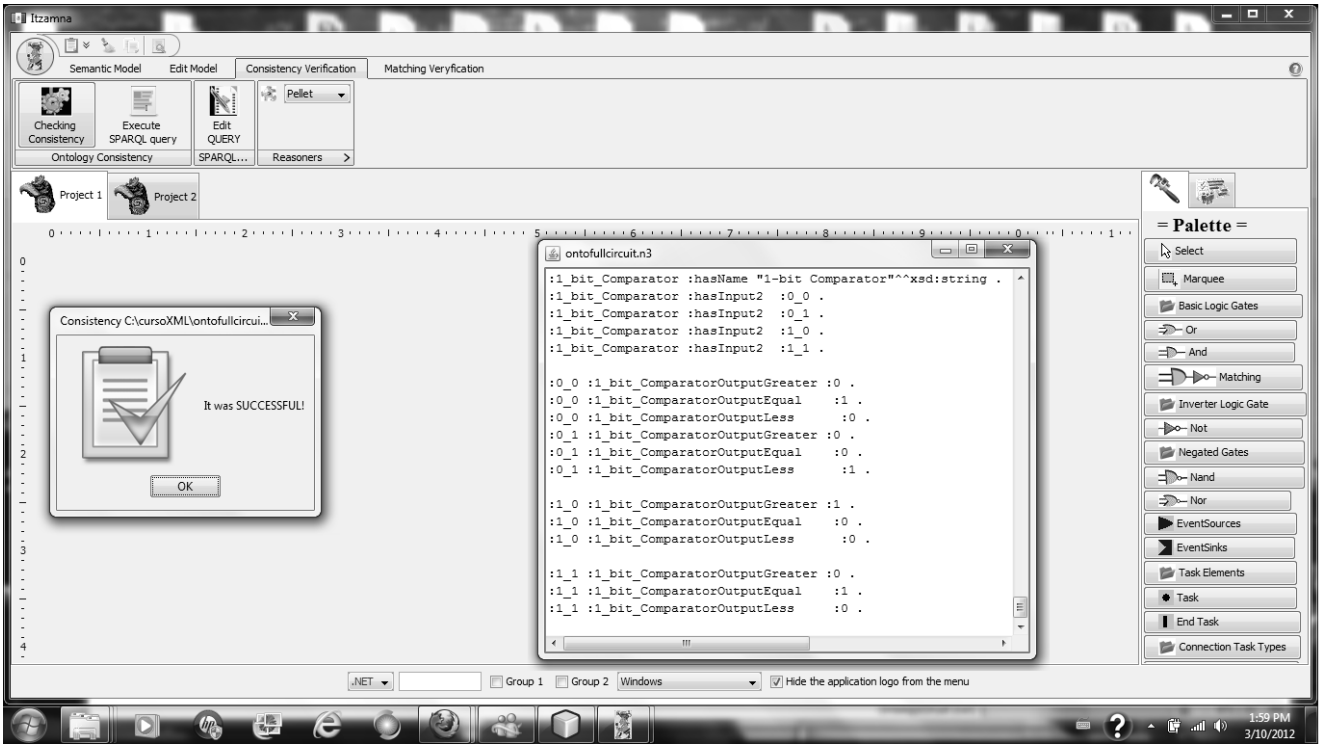

Figure 3. Consistency checking of the new circuit

the classes, properties and relations. Core Ontologies give more expressive meaning, maintains computability, do not require the validation of experts or apply a complex methodology for its construction. This core ontology for logic circuits increase the reuse of it and decrease the time in the development of future circuits. The use of an core ontology of logic circuits allowed us to validate the output of the 1-bit Comparator and verify the correct assembling of its gates using the Pellet reasoner and a SPARQL query with semantics in comparison with a classic SQL query. The queries on the ontology are simple and easy to do for all students whereas a classic SQL query in a database requires computational knowledge. In this paper we have presented a Semantic Web framework called Itzamna and described Semantic Web Techniques used for learning and teaching logic circuits.

7 Acknowledgments

This material is partly based upon work supported by the National Science Foundation under Grant No. OISE-0730065.

References

[1] F. E. Antoniou Grigoris and V. H. Frank. Introduction to semantic web ontology languages. 2005.

[2] E. Armstrong, J. Ball, S. Bodoff, D. B. Carson, I. Evans, K. Ganfield, D. Green, K. Haase, E. Jendrock, J. Jullion-ceccarelli, and G. Wielenga. The j2ee ^{TM}(tm) 1.4 tutorial for netbeans ^{TM}(tm) ide 4.1 for sun java system application server platform edition 8.1.

[3] K. Breitman, M. A. Casanova, and W. Truszkowski. *Semantic Web: Concepts, Technologies and Applications (NASA Monographs in Systems and Software Engineering)*. Springer-Verlag London, 2006.

[4] W. P. Davies John, Stunder Rudi. Semantic web technologies trens and research in ontology-based systems. 2006.

[5] S. Decker, M. Erdmann, D. Fensel, and R. Studer. *Ontobroker: Ontology based access to distributed and semi-structured information*. Citeseer, 1998.

[6] M. Doerr, J. Hunter, and C. Lagoze. Towards a core ontology for information integration. *Journal of Digital information*, 4(1), 2011.

[7] W. M. K. B. Duineveld A.J., Stoter R. and B. V.R. Wondertools? a comparative study of ontological engineering tools. 2000.

[8] N. F.Noy and D. L.McGuinness. Ontology development 101:a guide to creating your first ontology. March 2006.

[9] A. Gómez-Pérez, M. Fernández-López, and O. Corcho. Ontological engineering with examples from the areas of

knowledge management,e-commerce and the semantic web. 2003.

[10] T. Gruber. Ontolingua: A mechanism to support portable ontologies. pages KSL 91–66. 1992.

[11] T. Gruber. Toward principles for the design of ontologies used for knowledge sharing. pages 907–928. 1995.

[12] M. Gruninger and M. S. Fox. The role of competency questions in enterprise engineering. In *PROCEEDINGS OF THE IFIP WG5.7 WORKSHOP ON BENCHMARKING - THEORY AND PRACTICE*, 1994.

[13] N. Guarino. *Formal ontology in information systems: proceedings of the first international conference (FOIS'98), June 6-8, Trento, Italy*, volume 46. Ios Pr Inc, 1998.

[14] V. Haarslev and R. Müller. Racer system description. *Automated Reasoning*, pages 701–705, 2001.

[15] I. Horrocks, P. Patel-Schneider, and F. Van Harmelen. From shiq and rdf to owl: The making of a web ontology language. *Web semantics: science, services and agents on the World Wide Web*, 1(1):7–26, 2003.

[16] M. Hristozova and L. Sterling. An extreme method for developing lightweight ontologies. In *In Workshop on Ontologies in Agent Systems, 1st International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 2002.

[17] M. Hwang, H. Kong, and P. Kim. The design of the ontology retrieval system on the web. volume 3, pages 1815 –1818, feb. 2006.

[18] Java.net. Flamingo. http://java.net/projects/flamingo/, 2010.

[19] I. Jurisica, J. Mylopoulos, and E. Yu. Ontologies for knowledge management: an information systems perspective. volume 6, pages 380–401. Springer, 2004.

[20] A. Kiryakov, D. Ognyanov, and D. Manov. Owlim–a pragmatic semantic repository for owl. In *Web Information Systems Engineering–WISE 2005 Workshops*, pages 182–192. Springer, 2005.

[21] S. L. and M. B. Ontology evolution within ontology editors. volume 62, pages 53–62. September 2002.

[22] O. Lassila and R. Swick. Resource description framework (rdf) model and syntax. *World Wide Web Consortium, http://www. w3. org/TR/WD-rdf-syntax*, 1999.

[23] F.-M. Lesaffre and V. Pelletier. A business case of the use of ontologies for knowledge capitalization and exploitation.

[24] K. T. S. Michael C. Daconta, Leo J. Obrst. *The Semantic Web: A guide to the future of XML, Web Services and Knowledge Management*. Wiley Computer Publishing, Inc., 111 River Street Hoboken, NJ, jun. 2003.

[25] R. Mishra and S. Kumar. Semantic web reasoners and languages. *Artificial Intelligence Review*, pages 1–30, 2011.

[26] B. Motik and R. Studer. Kaon2–a scalable reasoning tool for the semantic web. In *Proceedings of the 2nd European Semantic Web Conference (ESWC05), Heraklion, Greece*, 2005.

[27] B. Parsia and E. Sirin. Pellet: An owl dl reasoner. In *Third International Semantic Web Conference-Poster*, 2004.

[28] J. Pérez, M. Arenas, and C. Gutierrez. Semantics and complexity of sparql. *The Semantic Web-ISWC 2006*, pages 30–43, 2006.

[29] T. Robal, T. Kann, and A. Kalja. An ontology-based intelligent learning object for teaching the basics of digital logic. In *Microelectronic Systems Education (MSE), 2011 IEEE International Conference on*, pages 106–107. IEEE, 2011.

[30] B. D. Rodriguez-Rocha, F. E. Castillo-Barrera, and H. Lopez-Padilla. Knowledge capitalization in the automotive industry using an ontology based on the iso/ts 16949 standard. volume 0, pages 100–106. IEEE Computer Society, Los Alamitos, CA, USA, sep. 2009.

[31] S. S., G.-P. A., D. W., R. M.-L., and N. N.F. Why evaluate ontology technologies? because it works! volume 19, pages 74–81. Jul-Aug 2004.

[32] S. S., S. R., S. H., and S. Y. Knowledge processes and ontologies. volume 16, pages 26–34. Jan-Feb 2001.

[33] S. Sosnovsky and T. Gavrilova. Development of educational ontology for c-programming. 2006.

[34] SWRL. Swrl:a semantic web rule language.

[35] B.-L. T., H. J., L. O., and Others. The semantic web. *Scientific American*, 284(5):34–43, 2001.

[36] N. L. J. Tolksdorf Robert, Bontas Paslaru Elena. A coordination model for the semantic web. pages 419–423. 2006.

[37] D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. *Automated Reasoning*, pages 292–297, 2006.

[38] W3C. http://www.w3.org/consortium/. 1994.

[39] W3C. Owl web ontology language, 1994.

[40] W3C. Semantic web activity: Semantic web activity state. 2002.