Proceedings of the International Symposium on Matching and Meaning, Michael Chan and Fiona McNeill (Eds.),
at the AISB 2010 convention, 29 March – 1 April 2010, De Montfort University, Leicester, UK

Evaluation of Error Taxonomy for OWL2

Naseer Ahmed Sajid [1], Dr. Muhammad Abdul Qadar [2]

Abstract. Ontology Evaluation is one of the most critical phases in ontology engineering. Applications depending upon ontology can have serious and appalling problems if ontology itself is infected with errors. Therefore, ontology should be consistent, complete and errors free. An error taxonomy using OWL constructs and axioms was developed to overcome the possibility of occurring errors in ontology. Recently, some new features have been proposed in OWL2 specification to overcome the limitation of expressive power of OWL. In this paper we evaluate the extended error taxonomy of OWL with the new features of OWL2. We also evaluate that while using OWL2 features, occurrence of possibility of new types of errors get reduced or not. On the basis of different examples for each error, we conclude that the error taxonomy is also valid for OWL2. This is because of the fact that every OWL ontology is a valid OWL2 ontology and every OWL2 ontology (excluding new features) is also a valid OWL ontology. The zero level circulatory errors can also be removed if we use the reflexive property and self restriction (local reflexive) property of OWL2. Otherwise, we have to generate a warning omission for zero level circulatory error.

1 Introduction

An explicit or formal specification of the domain is called ontology. It is a declarative representation which includes the terminology (or vocabulary) for referring to the terms in that domain and the logical statements that illustrate what the terms are, how they are related to each other, and how they can or cannot be related to each other. Ontology should represent formal specification about the domain concepts and relationships among them [5]. Ontologies therefore provide a vocabulary for representing and communicating knowledge about some domain and a set of relationships that hold among the terms in that vocabulary.

Like other component of a system, ontology has to go to through a repetitive process of refinement and evaluation during its development lifecycle. Ontology content evaluation is one of the critical phases of Ontology Engineering because if ontology itself is infected with errors then the applications dependent on it, may have to face some critical and appalling problems and ontology may not serve its purposes. Various types of research efforts have used to evaluate ontologies but besides the evaluation of ontologies, there is a need to evaluate the error taxonomy while development an ontologies. If ontology has some errors then there is possibility of errors in capturing and representing of different aspects of real world. So the evaluation of ontology ensures the quality of content and methodology.

Ontology evaluation includes ontology verification, validation and assessment. Ontology verification insures that its definition implement correctly, all the competency questions and requirements or functions correctly in the real world. Ontology validation insures the ontology definitions really model the real world for which the ontology was created. Ontology assessment is done by different types of users and application, judging the content of ontology from the user's point of view.

OWL is not enough because some ontologies use limited expressive power. It is not expressive enough as there is no user define data types [2, 3], no meta-modeling support [2, 3] and also limited property support [4].

OWL2 extends ontology web language by adding some new features and some extension in OWL. Users request these new small but useful features while they had problems in developing ontology. New features include extra syntactic sugar, additional property and qualified cardinality constructors, extended data type support, simple meta-modeling and extended annotations [1].

The remainder of this paper is structured as follows: the very next section is about OWL2 and its constructs and axioms. Error taxonomy is described in section 3. In section 4, we evaluate error taxonomy for OWL2 with examples. Paper is concluded in section 5.

2 OWL2

OWL2 is an ontology web language. It is an extension and revision of OWL developed by the web ontology working group of W3C. OWL2 is fully backwards compatible with OWL i.e. every OWL ontology is a valid OWL2 ontology and every OWL2 ontology (not using new features) is also a valid OWL ontology. OWL and OWL2 are intended to facilitate the sharing and development of ontology through web, with the purpose to easily access the contents of web to machines [6]. OWL2's ontologies provide classes, instances, properties and data values and stored in a semantic web documents. These ontologies can be written in RDF documents.

OWL2 provides some new features i.e. syntactic sugar, additional properties, database style keys, qualified cardinality constructors, and extended data types, simple meta-modeling and extended annotations. These new features increase the expressive power in OWL2. OWL2's profiles may meet better performance requirements and may be easier to implement. Main goals of OWL2 are to define profiles for OWL that are easier to implement and deploy as said above. It may cover all the important application areas and are easily understandable to non-expert users.

Faculty of Computer Science and Engineering, Muhammad Ali Jinnah University, Islamababad, Pakistan. [1] nasajid2005@gmail.com, [2] aqadir@jinnah.edu.pk

3 Extended Error Taxonomy

Errors taxonomy was classified by Gomez et al in [8], she formed error taxonomy for the assessment of ontologist. Ontology engineers used errors taxonomy for the development of errors free ontology. Keeping in mind of this errors taxonomy, developers forms a well define classification of concepts for sound semantic web vision. Errors taxonomy has basically three types of errors which are usually encountered by the ontologists. These are inconsistencies, incompleteness and redundancy [10]. There was a contribution in error taxonomy by many other researchers. Fahad et al [9], extend the error taxonomy for the evaluation of ontology. Errors Taxonomy with extended errors is shown in Figure 1.

4 Evaluation

In this section, we evaluate different types of errors which can be occurred while developing an ontology.

4.1. Inconsistency Errors

While reasoning from ontology, due to inconsistency and contradiction, three types of errors can be occurred. These errors are partition errors, circulatory errors and semantic inconsistency errors.

4.1.1. Partition Errors

Partition errors occur when we decompose a concept into various sub-concepts. Partition error occurs when a developer creates a common class which belongs to different disjoint classes or properties [10].
Concept classification can be defined in a disjoint/exhaustive decomposition.

Common classes in disjoint decompositions and partitions
There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. But other way round, there is an *DisjointUnion* axiom in OWL2 in which we can define a class and union of its disjoint classes. It is a modeling error and can be occur due to an un-experience developer. It can only be overcome by an experience person who is going to develop ontology.
Example:
If we define a MadCow class as a subclass of both Herbivorous and Carnivorous. Also if we define a MS Leading to Ph.D partition as a subclass of both Graduate and Post-Graduate partition.

Common instances in disjoint decompositions and partitions
These errors occur when a common instance belong to more than one class of a disjoint decomposition and partition.
Example:
If we define an instance ABC of both Herbivorous and Carnivorous classes. Also if we define a Naseer as an instance of both Graduate and Post-Graduate partition.

There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. It is a modeling error and can be occur due to an un-experience developer. It can only be reduce by an experience person who is going to develop ontology.

#	Type		Errors
1	Inconsistency	Partition	Common classes in disjoint decomposition and partitions of classes
2			Common instances in disjoint decomposition and partitions
3			External instances in exhaustive decomposition and partitions
4			Common property in disjoint decomposition and partitions of properties
5		Circulatory	Circulatory errors in class or in property hierarchy (Zero Level)
6			Circulatory errors in class or in property hierarchy (Non-Zero Level)
7		Semantic	More generalized concept as subclass
8			Domain violation by subclass
9			Disjoint Domain by subclass
10	Incompleteness	Partition	Sufficient knowledge omission
11			Disjoint knowledge omission among properties
12			Disjoint knowledge omission among classes
13			Exhaustive knowledge omission
14			Functional property omission
15			Inverse-Functional property omission
16			Incomplete concept classification
17	Redundancy		Identical formal definition of some classes
18			Identical formal definition of some properties
19			Identical formal definition of some instances
20		Grammatical	Redundancies of subclass of relations
21			Redundancies of sub-property of relations
22			Redundancies of instances of relations
23			Redundancies of disjoint of relations

Figure 1. Errors Taxonomy

External instances in exhaustive decompositions and partitions.
External instances in exhaustive decomposition and partition errors occur when there is at least one instance which belongs to the base concept but it does not belong to the sub-concepts.
Example:
If we can define a MadCow class as a subclass of Living animals but we have both Herbivorous and Carnivorous as exhaustive decomposition of Living Animals. Also if we can define a Ph.D Student class as subclass of Student but we only partitions of Graduate and UnderGraduate.

If a developer overlook some concept classification to disjoint with others then this may be occur. It is a modeling error and can be occur due to an un-experience developer and insufficient knowledge about that domain. There is no any axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. But other way round, there is an *DisjointUnion* axiom in OWL2 in which we can define a class and union of its disjoint classes. It can only be reduce by an experience person who is going to develop ontology.

Common property in disjoint decompositions and partitions in properties.

OWL allows to specify disjoint axioms between properties as well. Like common class, this error occurs when a common property is created in a disjoint decomposition of properties.

Example:

If we define hasAccess property as subProperty of both disjoint hasManagerPrivilages and hasClientPrivilages property.

In OWL2, there is no such axiom or feature which can reduce the possibility of occurrence of this type of error. It is a modeling error and can be occur due to an un-experience developer and insufficient knowledge about that domain. It can only be reduce by an experience person who is going to develop ontology.

4.1.2. Circulatory Errors

Circulatory errors occur when a developer or ontologist defines a class or property which is sub-class (or sub-property) or super-class (super-property) of itself at any level of hierarchy in the ontology [10].

Issues

There are some issues while applying the OWL2 new features. As in the literature it is already defined that there exist a circulatory zero level error e.g. Traveler is subclass-of itself, is a circulatory zero level error. OWL2 provides a reflexive property [7], which relates everything to itself. For examples, everybody has himself as a relative or every class is its own subclass or every person is a person. So reflexive property avoids these types of circular zero level error. But here it is again, if a developer or ontologist overlook this OWL2 new feature i.e. reflexive property and define such types of classes then circulatory zero level error may occur.

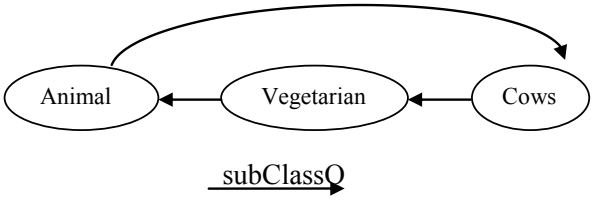

Figure 2a. Circulatory error in class hierarchy

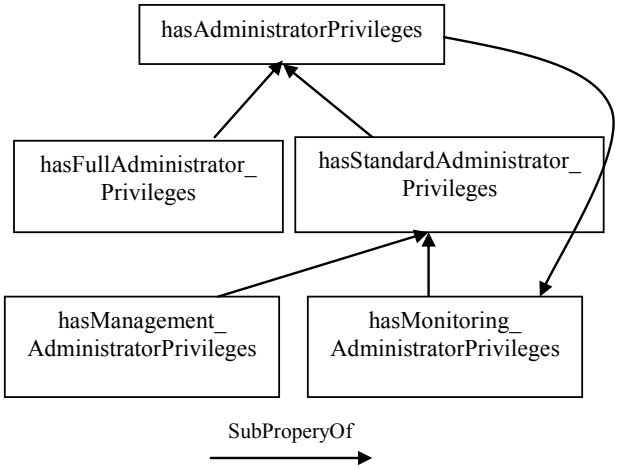

Figure 2b. Circulatory error in property hierarchy

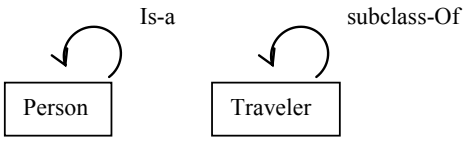

Figure 3. Reflexive

Similarly, OWL2 provides a self restriction or local reflexive property [7], which interlink instances or an individual might be link to itself. For example, when we say that all *NarcissisticPersons* love themselves or some persons killed themselves.

EquivalentClasses(:NarcissisticPerson ObjectHasSelf(:loves))

Self-restriction (local reflexivity): ∃ killed. Self

Or Kills (X,X)→ Suicide (X)
∃ Kills. Self ⊑ Suicide
and
Suicide (X)→ Kills (X,X)
Suicide ⊑ ∃ Kills. Self

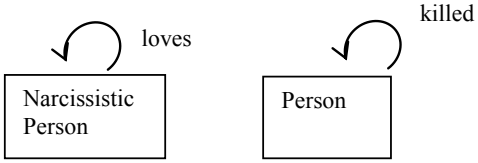

Figure 4. Self Restriction

Suicide is basically same as kills.

So self restriction or local reflexive property avoids these types of circular zero level error. But if a developer or ontologist overlook this OWL2 new feature i.e. self restriction or local reflexive property and define such types of classes or properties then circulatory zero level error may occur.

Hence to overcome above types of issues we have to generate warning for the ontologist or developer while developing ontology.

4.1.3. Semantic Errors

Semantic inconsistency errors occur when developer or ontologist create a class or subclass which violates the class hierarchy because that class does not actually belong to it [10].

Weaker Domain Specified by SubClass
These types of errors occur when ontologists specify subclass that possesses more general (weaker) domain as compared to the superclass domain [14], also when ontologists forget to specialize the concept going down in the hierarchy of concepts.
Example:

$$Person \sqcap \geq 5.hasChild.male$$
$$Person \sqcap \geq 2.hasChild.Person$$

If a developer is an un-experience person and has insufficient knowledge about that domain for which he/she is going to develop ontology, then this type of error occur. As there is also no such axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Domain Breach Specified by SubClass
These types of errors occur when ontologists specify subclass that overlaps other classes.

Example:

$$Person \sqcap hasSalary \geq 20{,}000$$
$$Person \sqcap (hasSalary \geq 10{,}000 \sqcap \leq 50{,}000)$$
$$Person \sqcap hasSalary \geq 40{,}000$$

Overlapping of subclasses may be specify by a developer if he/she is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology, then this type of error occur. As there is also no such axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Disjoint Domain Specified by SubClass
This error occur when ontologist specify a subclass that is disjoint with its domain of superclass.
Example:

$$Person \sqcap hasSalary \geq 20{,}000$$
$$Person \sqcap hasSalary \leq 15{,}000$$
$$Person \sqcap hasSalary \geq 40{,}000$$

OWL2 also has axiom of $DisjointWith$, it depends on the ontologist or developer to explicitly mention the disjointness among the classes and subclasses to avoid this type of errors. It can be also be occur due to a developer who is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology.

4.2. Incompleteness Errors

Sometime ontologist or developer while developing ontology, some information about the classification is overlooked by ontologist or developer. If some information is imprecisely model or possible information is missing about the decomposition, then some ambiguity and lack of reasoning arise due to this incompleteness. It is also possible when someone omitted concepts during the definition of the taxonomy and also if someone overlook some important information while classification of concepts.

4.2.1. Partition Errors

These types of errors can be occurred when some partitions of concepts is omitted. Some of these errors are as follow:

Disjoint Knowledge Omission among Classes
When a concept is classified into many subclasses and partition and knowledge about the disjointness of partition is omitted then this type of incompleteness error can be occur.
Example:
Qadir et al. [15] presented a case study of Access_Policy ontology in which disjoint knowledge omission between classes' administrator and user produces catastrophic results.
Example:
If we donot define AccessPolicy that userAccess and AdministratorAccess are disjoint subclasses of AccessPolicy.

OWL2 also has axiom of $DisjointWith$, it depends on the ontologist or developer to explicitly mention the disjointness among the classes and subclasses to avoid this type of errors. It can be also be occur due to a developer who is an un-experience and has insufficient knowledge about that domain for which he/she is going to develop ontology.

Disjoint Knowledge Omission among Properties
Like disjoint knowledge omission among classes, it is also significance importance of disjoint axioms between properties as well. So ontologist should check and specify knowledge between properties and avoid creating common instance among them.
Example:

Similarly, we have to mention $DisjointWith$ among the properties. OWL2 also provides a facility to explicitly mention the $DisjointWith$ among the properties.

Sufficient Knowledge Omission
Every concept should have necessary and sufficient knowledge so that application can reason about them. Necessary information provide by subclass of relation and sufficient information by other axioms of DL. Sufficient knowledge is not provided in the description of concepts, thus application can not differentiate between them. If there is only necessary description available and sufficient description of concepts is not available, due to these two reason sufficient knowledge Omission Error [12] can be occur.
Example:

$$Previlages$$
$$Administrator\_Full\_Previlages$$
$$Administrator\_Standard\_Previlages$$

There is no axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. It can reduce only by an experience person who has a sufficient knowledge about that domain for which he/she is going to model an ontology.

Exhaustive Knowledge Omission

This error occurs when ontologists do not follow the completeness constraint while decomposition of concept into subclasses and partitions [8].
Example:
In *Vehicle* ontology, *Bus* and *Motor-Cycle* classes are disjoint subclasses of *Vehicle* concept, but do not specified that whether or not this classification forms an exhaustive decomposition.

There is no axiom or feature available in OWL2 which can reduce the possibility of occurrence of this type of error. Ontologist or developer must have a good knowledge about the domain.

4.2.2. Functional and Inverse Functional Property Omission

By default owl takes multiple values for a given property. If a property takes just at most one value then it should be declared as functional property. e.g. age, height, User ID and Inverse Functional property is that for which two different objects cannot have the same value. e.g., NIC of Person, User ID of user.

$$< owl{:}\ DataProperty\ rdf{:}ID = "UserID" >$$
$$< rdfs{:}domain\ \ rdf{:}resource = \#User/>$$
$$< owl{:}range\ rdf{:}resource = \#int/>$$
$$</owl{:}\ DataProperty\ >$$

Needs restriction to take one and unique value

OWL2 provides both types of properties i.e. functional and inverse functional properties, but ontologists or developers have to mention it explicitly to avoid these types of errors.

4.2.3. Incomplete Concept Classification

This error occurs when ontologists overlook some of the concepts present in the domain while classification of particular concept [8].
Example:
If we want to make a *Living Animals* ontology and we ignore the *Omnivorous Animals*, then it is incomplete classification of concept *Living Animals*.
There is no axiom or feature available which can reduce the possibility of occurrence of this type of incompleteness errors. Experience ontologist or developer can reduce this error of incomplete concept classification.

4.3. Redundancy Errors

These types of errors occur when a particular information is derived from more than one classes, instances and their relationships. Detection of these types of redundancies is harder.

4.3.1. Identical Formal Definition of Classes/Instances

When Ontologist provides same formal definition of some classes but defines the different names for that classes [13].
Example:

$$User \sqsubseteq hasPrevilage\ AdministratorAccessPolicy$$
$$User' \sqsubseteq hasPrevilage\ AdministratorAccessPolicy$$

Both classes have same formal definition but with different names. Similarly, identical formal definition of instances cab also be occurred.

In OWL2, we can give same name to different types of entities with certain restrictions. E.g. The name used for a class or a datatype can also be used for an individual, object property, data property or annotation property. The name used for an individual can also be used for a class, datatype, object property, data property or annotation property

4.3.2. Identical Formal Definition of Properties

This type of error occur when ontologists specify multiple names which are semantically same for properties having same domain and range.
Example:

$$hasAccess\,(User, AccessPolicy)$$
$$hasPrevilage\,(User, AccessPolicy)$$

Here domain and range is same for given properties but both properties semantically are same.

Similarly, in OWL2, we can give same name to different types of entities with certain restrictions. E.g. The name used for an object or data or annotation property can also be used for an individual, class or datatype.

4.3.3. Grammatical Errors

There are some grammatical errors due to which redundancy may happen when developing taxonomies.

Redundancies in Subclass-of Relation
Redundancy of subclass-of relation errors take place when developer or ontologist make classes which have directly or indirectly subclass-of relations more than once [8].
Example:
If we want to make an *Vehicle* ontology and we define *Bus* as a subclass of *PassengerVehicle* and *MotorVehicle* creates redundancy as *PassengerVehicle* is already a subclass of *MotorVechicle*. Here indirect subclass of relation exists between *Bus* and *MotorVehicle* creating redundancy.

In OWL2, there is no such axiom or feature which can reduce the possibility of occurrence of this type of error. As it is a modeling error and can only reduce by an experience ontologist or developer who is going to develop an ontology for that particular domain. He/she should have as much knowledge as to require for a that particular domain.

Redundancies in Subproperty-of Relation
Redundancy of subproperty-of relation errors take place when developer or ontologist make data properties or object properties which have directly or indirectly subproperty relations more than once [8].
Example:
A datatype property *TelenorNo* is specified as subproperty of *MobileNo* and *ContactNo* as well, then redundancies of subproperty of relation exist.
Similarly, in this case, there is no axiom or feature in OWL2 which can reduce the possibility of occurrence of this type of error.

Redundancies in Instance-of Relation
When ontologists specify an instance x to a concept (C) and again specify x as instance of its (C) ancestor concepts [8], then this

type of error occurs. When a concept is specified as instance of a particular concept (C) then it is implicitly know as an instance of its parent concepts and does not need explicit description of instantiation to its super concepts.
Example:
If we specify instance *+923465100010* as an instance of *TelenorNo* and *ContactNo* classes, then it creates redundancy of instance of relation, as it is already defined that *TelenorNo* is a subclass of *ContactNo*. This explicit instance of relation between *923465100010* and *ContactNo* creates redundancy as *923465100010* is indirect instance of *ContactNo* as it is a superclass of *TelenorNo* class.
There is no restriction in OWL2 which can reduce this type of error. But ontologist or developer can do this, due to which this type of error will occur.

Redundancies in Disjoint-of Relation
If a concept is specified as disjoint with other concepts more than once [12], then this type of error is occurred. According to description logic rules [16], if a concept is disjoint with any concept then it is also disjoint with its sub-concepts. Only possibility of this error is that the concept is specified as disjoint with a concept but it was already specified that their parents were disjoint. Likewise redundancy of disjoint relation among property hierarchies exists in datatype property or object property hierarchies.

In OWL2, there are also tags for the disjoint of classes and properties. If we wrongly use these tags and mention the disjointness among classes and properties then this type of error will occur. Because we cannot specify the disjoint concept more than once.

5 Conclusion

We have discussed different types of errors and we have come to know that there are no such constructs or axioms or features in OWL2 specification which can remove the possibilities of occurring errors which is shown in Table 1. Hence all the errors are also valid in OWL2 except circulator zero level error. Although expressive power is increased by using OWL2 features. OWL2 also provide the punning (meta-modeling) concept which is used to give same name to different types of entities, with certain restrictions. Some issues are also occurred while using the OWL2 features i.e. due to omission of specifying reflexive property and self restriction or local reflexive property, there is a possibility of circulatory zero level errors. To overcome these issues, we have to generate warning for the developer or ontologist who is going to develop or model an ontology for that particular domain. For this purpose ontologist or developer should have sufficient knowledge to that particular domain for which he/she is going to develop an ontology.
We are also working to extend the error taxonomy by using the OWL2 constructs and axioms and will also generate warning for the omission of reflexive and self restriction property.

#	Errors	OWL	OWL2
1	Common classes in disjoint decomposition and partitions of classes	Valid	Valid
2	Common instances in disjoint decomposition and partitions	Valid	Valid
3	External instances in exhaustive decomposition and partitions	Valid	Valid
4	Common property in disjoint decomposition and partitions of properties	Valid	Valid
5	Circulatory errors in class or in property hierarchy (Zero Level)	Valid	Invalid
6	Circulatory errors in class or in property hierarchy (Non-Zero Level)	Valid	Valid
7	More generalized concept as subclass	Valid	Valid
8	Domain violation by subclass	Valid	Valid
9	Disjoint Domain by subclass	Valid	Valid
10	Sufficient knowledge omission	Valid	Valid
11	Disjoint knowledge omission among properties	Valid	Valid
12	Disjoint knowledge omission among classes	Valid	Valid
13	Exhaustive knowledge omission	Valid	Valid
14	Functional property omission	Valid	Valid
15	Inverse-Functional property omission	Valid	Valid
16	Incomplete concept classification	Valid	Valid
17	Identical formal definition of some classes	Valid	Valid
18	Identical formal definition of some properties	Valid	Valid
19	Identical formal definition of some instances	Valid	Valid
20	Redundancies of subclass of relations	Valid	Valid
21	Redundancies of sub-property of relations	Valid	Valid
22	Redundancies of instances of relations	Valid	Valid
23	Redundancies of disjoint of relations	Valid	Valid

Table 1. Comparison of OWL and OWL2 Error Taxonomy

References

[1]. "OWL2 Web Ontology Language: New Features and Rationale", W3C working group, http://www.w3.org/TR/2008/WD-owl2-new-features-20081202/
[2]. Jeff Z. Pan. Description Logics: Reasoning Support for the Semantic Web. Ph.D. Thesis, School of Computer Science, University of Manchester, 2004.
[3]. Jeff Z. Pan, Ian Horrocks and Guus Schreiber. OWL FA: A Metamodeling Extension of OWL DL. In Proc. of the International workshop on OWL: Experience and Directions (OWL-ED2005).
[4]. Jeff Z. Pan and Ian Horrocks. OWL-Eu: Adding Customised Datatypes into OWL (Extended version). In Journal of Web Semantics, 4(1): 29-39. 2006.
[5]. Antoniou, G., and Harmelen, F.V.,. *A Semantic Web Primer*. MIT Press Cambridge, ISBN 0-262-01210-3, 2004.
[6]. "OWL2 Web Ontology Language Document Overview", W3C working group, http://www.w3.org/TR/2009/WD-owl2-overview-20090327/
[7]. "OWL 2 Primer", W3C proposed recommendation 22 September, 2009, http://www.w3.org/TR/2009/PR-owl2-primer-20090922/
[8]. Gomez-Perez, A., Fernández-López, M., Corcho, O., *Ontological engineering: With examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web.* Springer, London, 2004.
[9]. Fahad, M., and Qadir, M.A. *A Framework for Ontology Evaluation.* Proceeding of 16th International Conference on Conceptual Structures (ICCS'08), Toulouse, France. Vol-354, pages 149-158, July 2008.
[10]. A. Gomez-Perez, M.F. Lopez, and O.C. Garcia, Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer ISBN:1-85253-55j-3, 2001.

[11]. M.A. Qadir, M. Fahad, S.A.H. Shah, Incompleteness Errors in Ontologies. Proc. of Intl GRC 07, USA. IEEE Computer Society. pp 279-282, 2007b.
[12]. W. Noshairwan, M.A. Qadir, M.A., M. Fahad, Sufficient Knowledge Omission error and Redundant Disjoint Relation in Ontology. InProc. 5th Atlantic Web Intelligence Conference June 25-27, France, 2007a.
[13]. Fahad. M., Qadir.M.A. and Noshairwan. W., "Ontological Errors-Inconsistency, Incompleteness, Redundency", In International Conference on Enterprise Information Systems (ICEIS) 2008, Barcelona, Spain.
[14]. Fahad, M., Qadir, M.A., and Noshairwan, W.,. *Semantic Inconsistency Errors in Ontologies*. Proceeding of International Conference on Granular Computing, Silicon Valley, USA, IEEE Computer Society, (2007b).
[15]. Qadir, M.A., Noshairwan, W., (2007c). *Warnings for Disjoint Knowledge Omission in Ontologies*. In Proc. of 2nd ICIW'07, Morne, Mauritius (2007), IEEE C.S, pp. 45
[16]. Nardi, D., et al. *The Description Logic Handbook: Theory, Implementation, and Applications*. ISBN: 9780521781763, 2002.
[17]. Grau. B., Horrocks. I., Motic. B., Parsia. B., Patel. P., Sattler. U., "OWL2: The Next Step for OWL", Web Semantics: Science, Services and Agents on the World Wide Web, Vol. 6, No. 4, pp 309-322, 2008.