A Framework for Ontology Evaluation

Muhammad Fahad, Muhammad Abdul Qadir

Center for distributed and Semantic Computing
Mohammad Ali Jinnah University, Islamabad, Pakistan,

mhd.fahad@gmail.com, aqadir@jinnah.edu.pk

Abstract. Mapping and merging of multiple ontologies to produce consistent, coherent and correct merged global ontology is an essential process to enable heterogeneous multi-vendors semantic-based systems to communicate with each other. To generate such a global ontology automatically, the individual ontologies must be free of (all types of) errors. We have observed that the present error classification does not include all the errors. This paper extends the existing error classification (Inconsistency, Incompleteness and Redundancy) and provides a discussion about the consequences of these errors. We highlight the problems that we faced while developing our DKP-OM, ontology merging system and explain how these errors became obstacles in efficient ontology merging process. It integrates the ontological errors and design anomalies for content evaluation of ontologies under one framework. This framework helps ontologists to build semantically correct ontology free from errors that enables effective and automatic ontology mapping and merging with lesser user intervention.

Keywords: Ontological Errors Taxonomy, Ontology Verification, Ontology Design Anomalies, Ontology Mapping and Merging, Semantic Web.

1 Introduction

To furnish the semantics for emerging semantic web, Ontologies should represent formal specification about the domain concepts and the relationships among them [1]. They have played a fundamental role for describing semantics of data not only in the emerging semantic web but also in traditional knowledge engineering, and act as a backbone in knowledge base systems and semantic web applications [10]. Like any other dependable component of a system, Ontology has to go through a repetitive process of refinement and evaluation during its development lifecycle before its integration in the semantic applications. Ontology content evaluation is one of the critical phases of Ontology Engineering because if ontology itself is contaminated with errors then the applications dependent on it, may have to face some critical and catastrophic problems and ontology may not serve its purpose [7].

Several approaches for evaluation of taxonomic knowledge on ontologies are contributed in the research literature. Ontologies can be evaluated by considering design principles [9,10,11], requirements and logical correctness of axioms, relations, instances, etc. Other approaches would be to evaluate ontologies in terms of their use in an application [18] and predictions from their results, comparison with a golden standard or source of data [13]. Considering design principles, Gomez formed error taxonomy for assistance in the ontology evaluation. Ontology engineers use that error taxonomy to build well-formed classification of concepts that enable better reasoning support for fulfillment of sound semantic web vision and to evaluate their ontologies in perspective of these errors. Besides taxonomic errors, there are some design anomalies which raise the issues of maintainability of ontologies [2].

This paper presents the ontological errors based on design principles for evaluation of ontologies. It provides the overview of ontological errors and design anomalies that reduces reasoning power and creates ambiguity while inferring from concepts. It shows our contribution in taxonomic errors that we experience while development of ontology merging system, *DKP-OM* [6]. Finally it integrates the design anomalies and taxonomic errors under one framework that helps practitioners, developers and ontologists to build well formed ontologies free from errors that serve their purposes, and develop tools for ontology evaluation for fulfilment of sound semantic web vision.

Rest of the paper is organized as follows: section 2 presents classification of ontological errors and design anomalies; section 3 contributes our identified ontological errors and extends the classes of errors formed by Gomez. Section 4 presents the related work of our domain. Section 5 concludes the paper.

2 Taxonomic Errors and Design Anomalies

Gomez-Perez [10,11] identified three main classes of taxonomic errors that might occur when modeling the conceptualization into taxonomies. The subsections elaborate each class of error made by Gomez.

2.1 Inconsistency Errors

There are mainly three types of errors that cause inconsistency and ambiguity in the ontology. These are Circulatory errors, Partition errors and Semantic inconsistency errors.

Circulatory errors: They occur when a class is defined as a subclass or superclass of itself at any level of hierarchy in the ontology. They can occur with distance 0, 1 or n, depending upon the number of relations involved when traversing the concept down the hierarchy of concepts until we get the same from where we started traversal. For example, circulatory error of distance 0 occurs when ontologist models *OddNumber* concept as subclass of *NaturalNumber* and *NaturalNumber* as subclass of *OddNumber*. As OWL ontologies provide constructs to form property hierarchies, so we have observed that circulatory errors in property hierarchies can occur.

Partition errors: There are mainly several ways of classification depending upon the type of decomposition of superclass into subclasses. When all the features of subclasses are independently described and subclasses do not overlap with each other then it leads to disjoint decomposition. When ontologists follow the completeness

constraint between the subclasses and the superclass, then it leads to a complete or exhaustive decomposition. The other can depend on both the disjoint and exhaustive decomposition. Three types of errors are:
Common instances and classes in disjoint decomposition and partitions: These errors occur when ontologists create the instances that belong to various disjoint subclasses or a common class as a subclass of disjoints classes. An example of former error is when ontologist decomposes the *Course* concept into disjoint subclasses *GradCourse* and *UndergradCourse*, and furthermore he classifies *CS6304* course as an instance of both disjoint classes. An example of later error is when ontologist decomposes the *NaturalNumber* concepts into disjoint subclasses *Odd* and *Even*, furthermore he classifies *Prime* number class as a subclass of both *Odd* and *Even* subclasses.
External instances in exhaustive decomposition and partitions: These errors occur when ontologists made an exhaustive decomposition or partition of a class into many subclasses but not all the instances of the base class belong to the subclasses, i.e., one or more instances of base class do not belong to any of the subclasses. For example ontologist decomposes *Accommodation* into *Hotel*, *House* and *Shelter* subclasses. This error occurs if he defines an instance *TrainStation* as an instance of the class *Accommodation*.

Semantic Inconsistency Errors: These errors occur when ontologists make an incorrect class hierarchy by classifying a concept as a subclass of a concept to which that concept does not really belong. For example he classifies the concept *SeaPlane* as a subclass of the concept *AirPlane*. Or the same might did when classifying instances. We find three main reasons that result incorrect semantic classification and classify the semantic inconsistency errors into three subclasses, explained in extension in taxonomic errors section.

2.2 Incompleteness Errors

Sometimes ontologists made the classification of concepts but overlook some of the important information about them. Such incompleteness often creates ambiguity and lacks reasoning mechanisms. The following subsections give the overview of incompleteness errors.

Incomplete Concept Classification: This error occurs when ontologists overlook some of the concepts present in the domain while classification of particular concept. For example ontologists classify concept *Location* into *CulturalLocation*, *MountainLocation*, and overlook other location types such as *BeachLocation*, *HistoricLocation*, etc.

Partition Errors: Gomez identified that sometimes ontologist omits important axioms or information about the classification of concept, reducing reasoning power and inferring mechanisms. He has identified two types of errors that cause incomplete partition errors to occur, that are:
Disjoint Knowledge Omission: This error occurs when ontologists classify the concept into many subclasses and partitions, but omits disjoint knowledge axiom between them. For example ontologist models the *BeachLocation*, *HistoricLocation*

and *MountainLocation* as subclasses of Location concept, but omits to model the disjoint knowledge axiom between subclasses. We developed the ontology of *Access_Policy*, where disjoint knowledge omission between User and Administrator causes catastrophic results [19], and provided the algorithm for identification of disjoint knowledge omission [16].

Due to significant importance of disjoint axiom between classes, OWL 1.1 allows to specify disjoint axioms between properties as well. So we also emphasis that ontologists should check and specify disjoint knowledge between properties, and avoid creating common instances between them.

Exhaustive knowledge Omission: This error occurs when ontologists do not follow the completeness constraint while decomposition of concept into subclasses and partitions. For example ontologist models the *BeachLocation, HistoricLocation* and *MountainLocation* as disjoint subclasses of Location concept, but does not specify that whether or not this classification forms an exhaustive decomposition.

2.3 Redundancy Errors

Redundancy occurs when particular information is inferred more than once from the relations, classes and instances found in ontology. The following are the types of redundancies that might be made when developing taxonomies.

Redundancies of SubclassOf, Subproperty-Of and InstanceOf relations: Redundancies of *SubclassOf* error occur when ontologists specify classes that have more than one *SubclassOf* relation directly or indirectly. Directly means that a *SubclassOf* relation exist between the same source and target classes. Indirectly means that a *SubclassOf* relations exist between a class and its indirect superclass of any level. For example ontologists specify *BeachLocation* as a subclass of *Location* and *Place*, and furthermore *Location* is defined as a *SubclassOf Place*. Here indirect *SubclassOf* relation exists between *BeachLocation* and *Place* creating redundancy. Likewise Redundancy of *SubpropertyOf* can exist while building property hierarchies. Redundancies of *InstanceOf* relation occur when ontologists specify instance *Swat* as an *InstanceOf Location* and *Place* classes, and it is already defined that *Location* is a subclass of *Place*. The explicit *InstancesOf* relation between *Swat* and *Place* creates redundancy as *Swat* is indirect instance of *Place* as *Place* is a superclass of *Location*.

Identical formal definition of classes, properties and instances: Identical formal definition of classes, properties or instances may occur when ontologist defines different (or same) names of two classes, properties or instances respectively, but provides the same formal definition.

2.4 Design Anomalies in Ontologies

Besides taxonomic errors, Baumeister and Seipel [2] identified some design anomalies that prohibit simplicity and maintainability of taxonomic structures with in ontology. These do not cause inaccurate reasoning about concepts, but point to problematic and badly designed areas in ontology. Identification and removal of these

anomalies should be necessary for improving the usability, and providing better maintainability of ontology.
Property Clumps: Datatype properties and Object properties that are associated with classes provide us powerful mechanisms for reasoning and inferring about concepts. Sometimes ontologists badly design ontology using repeatedly a group of properties in different class definitions. This repeated group of properties is called property clump and should be replaced by an abstract concept composing those properties in all the class definitions where this clump is used.
Chain of Inheritance: Ontology defines taxonomy of concepts and allows classifying concepts as *subClassOf* other concepts up to any level. When such hierarchy of inheritance is long enough and all classes have no appropriate descriptions in the hierarchy accept inherited child then that ontology suffers from *chain of inheritance*. For maintainability and simplicity, this chain of inheritance should be break-up into subhierarchies.
Lazy Concepts: Lazy concept is a leaf concept (or a property) in the taxonomy that never appears in the application and does not have any instances. Such concepts should be replaced with specialized or generalized concepts that occupy such instances and would be used in the application domain.
Lonely Disjoints: Sometimes ontologists need to modify the taxonomy of concepts and move concepts within the class hierarchy. Consider a scenario, where many disjoint siblings were created and later on a single sibling is moved to another place somewhere in the hierarchy, and ontologist forgets to delete the disjoint axiom between them. Such disjoint axioms should be removed from lonely disjoint concepts to enable better maintainability and reasoning support.

3 Extensions in Taxonomic Errors

We have identified several ontological errors [7,15,16,19,20] while evaluating taxonomic knowledge on ontologies and knowledge based systems, and extended the main three classes of Taxonomy evaluation, i.e., Inconsistency, Incompleteness and Redundancy. Some of these are experienced while developing *DKP-OM*: Disjoint Knowledge Preserver based Ontology Merger [6], a solution we provide for effective ontology merging. The subsections present our identified ontological errors.

3.1 Semantic Inconsistency Errors

There are mainly three reasons due to which incorrect semantic classification originates [7]. According to these reasons, we categorize Semantic inconsistency errors into three subclasses. These subclasses can be used as a check list for class hierarchy evaluation and help in building well-formed class hierarchy to provide better interpretation of concepts.
Weaker domain specified by subclass error: When classes that represent the larger domain are kept subclasses of concept that possess smaller domain then such an error might occur. For example ontologist classifies *UniversityMember, AcademicStaff, AdminStaff* and *LabStaff* concepts as a subclass of a concept *Staff* superclass. Here the

semantic inconsistency occurs as he classified more generalized concept *UniversityMember* as subclass of the concept *Staff*. A subclass should always specializes (subsumed by) the superclass concept's properties by specifying stronger domain and make the super concept's domain narrower.

Domain breach specified by subclass error: Subconcepts should possess all the features of the parent concept and should not violate any feature of their parent concept in their own domain. Superclass domain breach occurs when concepts treated as subclasses add more features that are not present in superclass but the additional features are violating some features of their superclasses. For example consider a *Pizza* class hierarchy where ontologist classifies concept *VegetarianPizza* as a subclass concept of *Pizza*. Furthermore he classifies *ChinesePizza* and *ItalianPizza* concepts as the subclasses of the concept *VegetarianPizza*. Semantic Inconsistency arises as the definition of *ChinesePizza* allows having any toppings made from boiled vegetables and any kind of meat.

Disjoint domain specified by subclass error: When ontologists specify disjoint domain concepts as subclasses of a concept that occupies a different domain. For example he classifies concepts *Drink* and *Burger* as subclasses of *EatableThing* concept. None of the features of *Drink* match with superclass concept *EatableThing* i.e. they belong to disjoint domains.

These semantic inconsistency errors can be applied same to the instances of superclass and subclasses to whether their conformance with each other.

3.2 Extension in Incompleteness Errors

For powerful reasoning and enhanced inference, OWL ontology provides some tags that can be associated with properties of classes [17]. OWL functional and inverse-functional tags associated with properties indicate how many times a domain concept can be associated with range concept via a property. Sometimes ontologists do not give significance to these property tags and do not declare datatype or object properties as functional or inverse-functional. As a result machine can not reason about a property effectively leading to serious complications [20].

Functional Property Omission (FPO) for single valued property: According to Ontology Definition Metamodel [17], when there is only one value for a given subject then that property needs to be declared as functional. The tag *Functional* can be associated with both the object properties and datatype properties. For example *hasBlood_Group* as an object property between *Person* and *Blood_Group* is an example of functional object property. Every subject *Person* belongs to only one type of *BloodGroup*, so *hasBlood_Group* property should be tagged as functional so that person should be associated with one blood group. Likewise functional datatype properties allow only one range R for each domain instance D. Ignoring Functional tag allows property to have more than one values leading to inconsistency. One of the main reason for such inconsistency is that ontologist has ignored that OWL ontology by default supports multi-values for datatype property and object property.

Inverse-Functional Property Omission (IFPO) for a unique valued property: According to Ontology Definition Metamodel [17], inverse-functional property of the object determines the subject uniquely, i.e. it acts like a unique key in databases. This means that if we state P as an *owl InverseFunctionalProperty*, then this restricts that for a single instance there can only be a value x, i.e. there cannot exist two different instances y and z such that both pairs (y, x) and (z, x) are valid instances of P. In OWL Full, datatype property can be tagged as inverse-functional property because datatype property is a subclass of object property. But in OWL DL datatype property can not be tagged as inverse-functional property because object properties and datatype properties are disjoint. An example of inverse object property is *National_SecurityNo* that belong to the *Person* as it uniquely identifies the *Person*. Ignoring inverse-functional tag with the property *National_SecurityNo* creates inconsistency within the ontology due to incomplete specification of concept. We consider such lack of information as an error, because such ignorance leads machine not to infer and reason about concepts uniquely.

Sufficient knowledge Omission Error (SKO): Ontology comprises concepts and properties that can be arranged in hierarchies. These concepts in hierarchies should posses some features so that inference engine can distinguish them appropriately. According to principles of Description Logic, there should be *Necessary description* and *Sufficient description* associated with each concept [14]. *Necessary description* rules define the basic criteria by which new concept is formed by subclass of relation, and *Sufficient description* defines the concept in terms of another concepts like its self description by using intersection, union, complement or restriction axioms in OWL [15]. Sometimes during ontology designing, ontologists define the concepts but don't provide their *Sufficient descriptions*. As a result, machine can't reason about them properly and cannot use them effectively to achieve the goals of semantic web.

Finding incompleteness in ontologies automatically is a difficult task. One of the possible ways to detect such incompleteness errors is to evaluate ontology on test data [4] (valid and invalid both) that can be generated according to tester's domain knowledge [22], experience with similar concepts and information about soft spots of ontology.

3.3 Extension in Redundancy Errors

While detecting disjoint knowledge omission in ontology and generating warnings on its omission [15], we detect redundancy of disjoint relation in ontologies. The following subsection provides detail on it.

Redundancy of Disjoint Relation (RDR) Error: Redundancy of Disjoint Relation occurs when the concept is explicitly defined as disjoint with other concepts more than once (Noshairwan, 2007a). By Description Logic rules [14], if a concept is disjoint with any concept then it is also disjoint with its sub concepts. The one possible way of occurrence of RDR is that the concept is explicitly defined as disjoint with parent concept and also with its child concept. For an example, concept Male is defined as disjoint with Female and also with sub concepts of Female. This type of redundancy can occur due to direct disjointness (directly disjoint) and indirect disjointness (concept is disjoint with other because its parent is disjoint with it).

4 Related Work

There are many other approaches for ontology evaluation but still there is a big gap which needs to be filled for sound semantic web ontologies. The standard ontology evaluation approach by Maedche and Staab [13] is to compare ontology with gold standard ontology for evaluating lexical and vocabulary level of ontology. Besides comparison with gold standard, Brewster et al. [4] gave the corpus or data driven ontology evaluation approach. Comparison of ontology with the corpus or data of the domain knowledge provides a measure of the fit between them; and highlights the terms that are present/absent in ontology and corpus. Context level evaluation approach takes into account the larger collection of ontologies as a reference for evaluation of particular ontology [22]. The library of ontologies or the context for evaluation provided by the knowledge engineer acts as reference to follow. Other approaches of ontology evaluation would be to observe the results of application or task where this ontology is being used. Prozel and Malanka [18] proposed the task-based approach for ontology evaluation but could not be so much effective, as ontology acts only a backbone and several other issues of task itself can generate bad results. Burton-Jones [5] defined a semiotic metrics based on different criteria for ontology assessment for syntactic and lexical/vocabulary evaluation. Likewise Fox el al. [8] made a set of parameters but these are more useful for manual assessment of quality of ontology. These ontology evaluation approaches are useful in different applications, scenarios and environments [3] and the choice of a suitable methodology should be adopted according to the ontology usage.

5 Conclusion

Ontology driven architecture has revolutionized the inference system by allowing interoperability between heterogeneous multi-vendors systems. We have identified that accurate ontologies free from errors enable more interoperability, improve the accuracy of ontology mapping and merging and lessen human intervention during this process. We have discussed existing ontological errors, and identified newer types of errors present in ontologies. We have concluded that without identification and removal of these errors the most desirable goal of ontology mapping and merging could not be achieved. We have integrated the overall work about ontology evaluation based on design principles and anomalies under one framework. This framework acts as control mechanism that helps ontologist to build accurate ontologies that serve best for the desired applications, provide better reasoning support, lessen user intervention in efficient ontology merging and combined use of independently developed online ontologies can be made possible.

References

1. Antoniou, G., and Harmelen, F.V. 2004. A Semantic Web Primer. MIT Press Cambridge, ISBN 0-262-01210-3
2. Baumeister, J., and Seipel, D.S. 2005. Owls–Design Anomalies in Ontologies", 18th Intl. Florida Artificial Intelligence Research Society Conference (FLAIRS), pp 251-220
3. Brank J. et al. 2005. A Survey of Ontology Evaluation Techniques. Published in multi-conference IS 2005, Ljubljana, Slovenia SIKDD.
4. Brewster, C. et al. 2004. Data driven ontology evaluation. Proceedings of Intl. Conf. on Language Resources and Evaluation, Lisbon.
5. Burton-Jones, A., et al. 2004. A semiotic metrics suite for assessing the quality of ontologies. Data and Knowledge Engineering.
6. Fahad, M., Qadir, M. A., Noshairwan, M., W., Iftikhar, N. 2007a. DKP-OM: A Semantic Based Ontology Merger. In Proc. 3rd International conference on Semantic Technologies, I-Semantics 5-7 September 2007, Journal of Universal Computer Science (J.UCS).
7. Fahad, M., Qadir, M.A., Noshairwan, W. 2007b. Semantic Inconsistency Errors in Ontologies. Proc. of GRC 07, Silicon Valley USA. IEEE CS. pp 283-286
8. Fox, M. S., et al. 1998. An organization ontology for enterprise modelling. In: M. Prietula et al., Simulating organizations, MIT Press.
9. Gomez-Perez, A. 1994. Some ideas and examples to evaluate ontologies. KSL, Stanford University.
10. Gomez-Perez, A., Lopez, M.F, and Garcia, O.C. 2001. Ontological Engineering: With Examples from the Areas of Knowledge Management, E-Commerce and the Semantic Web. Springer ISBN:1-85253-55j-3
11. Gomez-Perez, A., et al. 1999. Evaluation of Taxonomic Knowledge on Ontologies and Knowledge-Based Systems. Intl. Workshop on Knowledge Acquisition, Modeling and Management.
12. Jelmini, C., and M-Maillet S. 2004. OWL-based reasoning with retractable inference", In RIAO Conference Proceedings 2004.
13. Maedche, A., Staab, S. 2002. Measuring similarity betwe- en ontologies. Proc. CIKM 2002. LNAI vol. 2473.
14. Nardi, D. et al. 2000. The Description Logic Handbook: Theory, Implementation, and Applications. Noshairwan, W., Qadir, M.A., Fahad, M. 2007a.
15. Sufficient Knowledge Omission error and Redundant Disjoint Relation in Ontology. InProc. 5th Atlantic Web Intelligence Conference June 25-27, 2007 - Fontainebleau, France
16. Noshairwan, W., and Qadir M.A. 2007b. Algorithms to Warn Against Incompleteness Errors in Ontology Evaluation. 1st AISPC Jan 2007.
17. Ontology Definition Metamodel 2005. Second Revised Submission to OMG/RDF
18. Porzel, R., Malaka, R., 2004. A task-based approach for ontology evaluation. ECAI 2004 Workshop Ont. Learning and Population.
19. Qadir, M.A., Noshairwan, W. 2007a. Warnings for Disjoint Knowledge Omission in Ontologies. Second International Conference on internet and Web Applications and Services (ICIW07). IEEE, p. 45
20. Qadir, M.A., Fahad, M., Shah, S.A.H., 2007b. Incompleteness Errors in Ontologies. Proc. of Intl GRC 07, USA. IEEE Computer Society. pp 279-282
21. Qadir, M.A., Fahad, M., Noshairwan, W. 2007c. On Conceptualization Mismatches in Ontologies. Proc. of GRC 07, USA. IEEE CS. pp 275-279
22. Supekar, K. 2005. A peer-review approach for ontology evaluation. Proc. 8th Intl. Protégé Conference, Madrid, Spain, July 18–21, 2005.