Desirable Ontologies for the Construction of Semantic Applications

R.Pavan Kartheek[1], M. Gamya[2], Dr. Sk. Nazeer[3]

Department of Computer Science
Bapatla Engineering College (Autonomous)
Bapatla,India

***Abstract*-**The intended goal of semantic web is to provide the search results to the user with at most accuracy and good precision. To make possible, the object of semantic web is to add semantics to the existing information on the Web using semantic web languages. These web languages have been to express detail information of the content present on the web with help of Ontologies. Ontology is expressed in a knowledge representation language, which provides a formal frame of semantics. Therefore we provide a brief explanation of semantic web languages in which some of them uses description logic and frames as basis. These semantic languages used in construction and understanding of ontologies clearly. The goal of this paper is to provide a brief survey of state-of-the-art ontology languages which are used to express ontology over the Web, a basic understanding of ontologies and how the ontologies are constructed.

Keywords*-Ontology, Semantic Web, web languages, description languages, knowledge representation language, description logic, Frames.*

*****

1. INTRODUCTION

The World Wide Web is a repository of information that is structured for human readers and is not suitable for machines in formalizing the meaning of content. The information on the web is organised using HTML, which is used for presentation. This situation is somewhat alleviated by the Extensible Markup Language (XML). However, although XML Document Type Declarations (DTDs) can specify the grammar of markup languages, there are no facilities for understanding the semantics of the content by the machine. To create a web with semantics, one must extend XML with features of knowledge representation (KR) languages. Using these languages, Ontologies and intelligent agents are build. An approach to make the information accessible to intelligent agents though ontologies is semantic web. The aim of ontologies is to capture domain knowledge and their role is to create semantics explicitly in more generic way. They provide interoperability between web applications and make the web content machine understandable.

The mechanism of semantic web search is annotating semantics to the existing web pages on web using one of the semantic web languages SHOE, RDF/RDF(S), OIL, DAML+OIL and OWL. And this content is accessible by intelligent agents or semantic search engines to retrieve content with high precision and accuracy. All semantic languages are based on XML syntax, but they have

different terminologies and expressions. Indeed, some of these languages have the ability to represent certain logical relations which others do not. Because some languages have greater expressive power than others, their selection for representing ontologies is based mainly on what the ontology represents or what it will be used for. In other words, different kinds of ontological knowledge-based applications need different language facilitators to enable reasoning on ontology data. These description languages provide richer constructors for forming complex class expressions and axioms.

This paper provides the overview of Ontology and its structure along with SHOE [2], RDF/RDF(S), OIL, DAML+OIL and OWL.

2. ONTOLOGY

Ontology: "Ontology [4] [6] [7] is an explicit, machine readable specification of a shared conceptualization."

Ontologies are the essential elements in many applications. They are used in agent systems, knowledge management systems, and e-commerce platforms. They can integrate intelligent information to the content and provide semantic-based access to the Internet. Using ontologies we can extract information from texts conceptually which makes major change in current Web contents. This change is leading to the third generation of the Web—known as the Semantic Web—which has been defined as "the conceptual structuring of the Web in an explicit machine-readable way". Ontology provides domain knowledge and not the structure of a data container.

Knowledge in ontologies organised using five kinds of components: concepts, relations, functions, axioms and instances. Concepts in the ontology are usually structured in taxonomies. These are widely used to organise ontological knowledge in the domain using generalisation/specialisation relationships through which simple/multiple inheritance could be applied.

There are different languages to support and express the ontologies completely. These languages fall generally into three categories: Ontology defined using natural language, frame-based languages used to build the structure of

 ISSN: 2321-8169
4047 - 4055

Ontologies based on explicit statements of class and slot, and the languages based on logic, such as Description Logics.

The main advantages of ontologies are reuse of domain knowledge, make domain assumptions explicit, separate domain knowledge from the operational knowledge and to analyze domain knowledge.

There are three different types of ontologies.Those are,

2.1 KR Ontologies

Knowledge representation (KR) ontology [8] gathers the modelling primitives used to formalize knowledge in a KR paradigm. Examples of such primitives are classes, relations, attributes, etc.

2.2 Top-level ontologies

Describes very general concepts that are common across the domains and give general notions under which all the terms in existing ontologies should be linked to. Sometimes top-level ontologies [8] are used to build domain ontologies, but often these are built first and then linked to upper-level ontologies.The following characteristics are identified as desirable in a top-level ontology are universal and articulate.

2.3 Linguistic Ontologies

The purpose of this type of ontology is to describe semantic constructs rather than to model a specific domain. They offer quite a heterogeneous amount of resources, used mostly in natural language processing. The main characteristic of these ontologies is that they are bound to the semantics of grammatical units. In some of the ontologies there is a one-to-one mapping between concepts and words in a natural language while in others many concepts may not map to any word in a language or may map to more than one in the same language.

3. WAYS IN CONSTRUCTING ONTOLOGY

3.1 Description Logic:

Description Logic (DLs) [6]attempt to find a fragment of first-order logic with high expressive power which still has a decidable and efficient inference procedure.

These are used to describe ontology knowledge in terms of concepts and restrictions on roles and also used to generate taxonomies automatically among concepts.

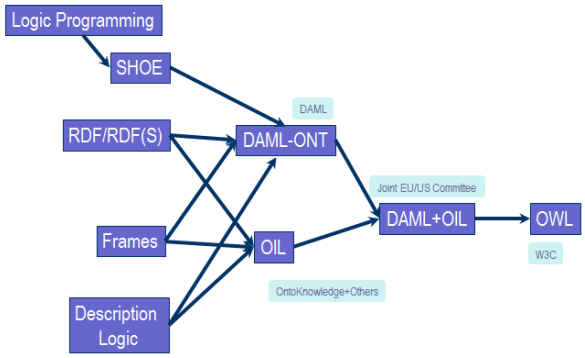
DLs has a feature that classes(called concepts) can be defined in terms of descriptions that are used to specify properties, which are used to identify objects that must belong to these classes(concepts).

A language can be used to implement descriptions include restrictions on roles(called binary relationships).

3.2 Frame based Systems:

The central modelling primitives of predicate logic are predicates. Frame-based [6] and object-oriented approaches take a different point of view. Their central modelling primitives are classes (i.e., frames) with certain properties called attributes. These attributes do not have a global scope but are only applicable to the classes they are defined for (they are typed) and the ”same” attribute (i.e., the same attribute name) may be associated with different value restrictions when defined for different classes. A frame provides a certain context for modelling one aspect of a domain.

4. EVOLUTION OF ONTOLOGY LANGUAGES



4.1 SHOE (Simple HTML Ontology Extensions)

SHOE is semantic web language extends HTML with a set of knowledge oriented tags that uses in ontology construction, unlike HTML tags, provide structure for knowledge acquisition as opposed to information presentation. It enables web page authors to annotate their web documents with ontologies which associates meaning to the content by making each web page commit to one or more ontologies. These ontologies permit the discovery of implicit knowledge through the use of taxonomies and inference rules, allowing information providers to encode only the necessary information on their web pages, and to use the level of detail that is appropriate to the context.

The SHOE ontologies made available to all in order to extend existing ontologies which are already developed, forming a hierarchy with the most general ontologies at the top and the more specific ones at the bottom. Also the authors of web pages develop ontologies with their own familiar vocabulary which are not common among all the related content WebPages. To promote uniformity and Interoperability sharing and reuse of ontologies is suggested by SHOE group.

Ontologies inherit all of the components present in their ancestors. In SHOE [2], an ontology is an ISA hierarchy of classes (called categories), plus a set of atomic relations between them, and inferential rules in the form of simplified horn clauses. These ontologies provide definitions of classes and properties (called categories and relations in SHOE). SHOE is not having any predefined ontologies, using this language we specify categories, relationships, attributes, inferences in construction of any ontology.

SHOE is not based on description logic, it is based on datalog.

The syntax of datalog is basically Prolog without function symbols. SHOE does not have as rich expressions for defining classes. SHOE also allow intelligent agents to make automatic inferences about the data, provides a hierarchical categorization scheme, and a sophisticated ontology mechanism designed specifically for the web needs.

SHOE uses the following additional tags as an extension to HTML for the definition of ontologies:

<ONTOLOGY>,</ONTOLOGY>,<USE-ONTOLOGY>,<DEF-CATEGORY>,<DEF-RELATION>, </DEF-RELATION>,<DEF-ARG>,<DEF-RENAME>,<DEF-CONSTANT>,<DEF-TYPE, DEF-INFERENCE>, </DEF-INFERENCE>,<INF-IF>,< /INF-IF>,<INF-THEN>,</INF-THEN>, <COMPARISON>,</COMPARISON>, <CATEGORY, RELATION>, </RELATION>, <ARG>, <INSTANCE>, </INSTANCE>.

4.2 RDF (Resource Description Framework)

RDF [4] [7] is the basic building block for supporting the Semantic Web. It is an XML-based language for describing information contained in a Web resource. RDF is about metadata. It is a language recommended by W3C which describes any resource independent of any domain and provides a basis for coding, exchanging, and reusing structured metadata. RDF is machine-understandable. Machines can do useful operations with the knowledge expressed in it. RDF allows interoperability among applications exchanging machine understandable information on the Web.

4.2.1 The elements in RDF:

- **Resource** is anything that is described in RDF expressions. Ex: web page, website, part of web page or any real world object.
- **Property** is a resource that has a name and can be used as a property; i.e., it can be used to describe specific aspect, characteristic, attribute, or relation of the given resource.
- **Statement** is used to describe the properties of a resource. It is in the format: (resource) subject + (property) predicate + (property value) object.

The value of property can be a string literal or a resource. Therefore, in general, an RDF statement indicates that a resource (the subject) is linked to another resource (the object) via an arc labelled by a relation (the predicate).

It can be interpreted as follows:<Subject> has a property <predicate>, whose value is <object>

The subject, predicate and object is identified with URI if the object is a resource. Knowledge is expressed in statements in the form of subject, predicate and object the order is never changed. The subject, object and predicate together call as RDF triple.

The RDF vocabulary set:RDF, Description, ID, about, parseType, resource, li, nodeID, datatype, Seq, Bag, Alt, Statement, Property, XMLLiteral, List, subject, predicate, object, type, value, first, rest_n.

4.2.2 Merits:

- To make XML more interoperable RDF format is used.
- Characterize the elements to a specific type (class) along with their properties.
- RDF can be used to describe resources in a structured way that machines can process.
- RDF promotes the use of standardized vocabularies.
- You can use the RDF tools to apply inference to the data.
- It positions your data for the Semantic Web!

4.2.3 Limitations:

- The RDF format constrains you on how you design your XML.
- RDF uses namespaces to uniquely identify types (classes), properties, and resources. Thus, you must have a solid understanding of namespaces.

4.3 RDF(S) (Resource Description Framework Schema)

RDFS [4] [7] is an extension to RDF. It is a language one can use to create a vocabulary for describing classes, subclasses, and properties of RDF resources. It adds semantics to RDF predicates and resources. RDFS is used to define RDF vocabularies. The root class of everything is rdfs: resource. The RDFS file starts with RDF and RDFS namespaces where the keywords for RDF and RDFS are defined.

4.3.1 The vocabulary set of RDFS:

- **Core classes**: rdfs:resource, rdf:property, rdfs:class, rdfs:datatype

 ISSN: 2321-8169
4047 - 4055

- **Core properties**: rdfs:subClassOf, rdfs:subPropertyOf
- **Core constraints**: rdfs:range, rdfs:domain

4.3.2 Core Classes: The elements in this class could be called fundamental concepts because they are used to describe most classes and their properties.

- **rdfs:resource**:Anything that is described with RDF expressions is a resource which an instance of rdfs:resource. These resources are identified with URIs.
- **rdf:property :** It is used to represent that subset of RDF resources called properties.
- **rdfs:class :** It is used to describe class which is an instance of rdfs:resource represents anything. It also uses rdf:ID to provide name of the class.
- **rdfs:datatype :** It is used to indicate the data type.

4.3.3 Core Properties

These properties are in fact considered as an instances of the class rdf:property. They are used to provide a mechanism for expressing relationships between classes and super-classes or between classes and their instances.

- **rdfs:subClassOf :**This property is a transitive relation used to identify a relation between classes as sub/supersets. On using this property classes are organised into subset hierarchy. If any class which not defined with this property is assumed as direct subclass of rdfs:resource.
- **rdfs:subPropertyOf :** This defines any property used to represent a relation between resources. This kind of property is a specialisation relation. rdfs:subPropertyOf is applied to properties to denote that one property is a subset or specialisation of another.
- **rdfs:range&rdfs:domain :**rdfs:range is used to declare that the values of a property are instances of one or more classes. rdfs:domain is used to specify which class the property being defined can be used with one or more classes. When we use rdfs:range and rdfs:domain properties, rdf:resource must be used along with them. Both properties are optional.

4.3.4 Merits:

- RDFS is a *loose* collection of relations.
- Applications may do “database”- like search.
- Easy to combine relations in one big collection and provides integration of heterogeneous information.

4.3.5 Limitations:

- RDFS too weak to describe resources in sufficient detail like No localised range and domain constraints, No existence/cardinality constraints, No transitive, inverse or symmetrical properties.

Table1 4.3:Property Descriptions of RDF(S)

Property name	Domain	Range
rdf:type	rdfs:Resource	rdfs:Class
rdf:subject	rdf:Statement	rdfs:Resource
rdf:predicate	rdf:Statement	rdf:Property
rdf:object	rdf:Statement	rdfs:Resource
rdf:value	rdfs:Resource	rdfs:Resource
rdf:first	rdf:List	rdfs:Resource
rdf:rest	rdf:List	rdf:List
rdfs:subClassOf	rdfs:Class	rdfs:Class
rdfs:subPropertyOf	rdf:Property	rdf:Property
rdfs:comment	rdfs:Resource	rdfs:Literal
rdfs:label	rdfs:Resource	rdfs:Literal
rdfs:seeAlso	rdfs:Resource	rdfs:Resource
rdfs:isDefinedBy	rdfs:Resource	rdfs:Resource
rdfs:member	rdfs:Resource	rdfs:Resource
rdfs:domain	rdf:Property	rdfs:Class
rdfs:range	rdf:Property	rdfs:Class

4.4 OIL (Ontology Interchange Language)

OIL [6] [7] is developed in the Onto Knowledge project permits semantic interoperability between Web resources. Ontology Interchange Language is based on description logic and frames.

OIL is a common core ontology language which is used for ontology design, sharing – exchange, integration and verification. The sharing and exchange can be seen to have two dimensions. The unequivocal sharing of semantics so that when the ontology is deployed it can be interpreted in a consistent manner. Ensuring that when the ontology is viewed by an agent (in particular here a person) other than the author, the intention of the author is clear. Its syntax and semantics are based on existing proposals OKBC, XOL, and RDF(S), providing modelling primitives commonly used in frame-based approaches to ontological engineering (concepts, taxonomies of concepts, relations, and so on), and formal semantics and reasoning support found in description logic approaches (a subset of first order logic that maintains a high expressive power, together with decidability and an efficient inference mechanism). It is built on RDF and RDF/S, this language provides modelling primitives used in frame based and Description Logic oriented Ontologies. It is standard for describing and exchanging ontologies.

ISSN: 2321-8169
4047 - 4055

OIL is layered language model consisting of *Core OIL, Standard OIL, Instance OIL, and Heavy OIL.* RDFS is largely combine in Core OIL layer due to this RDFS agents can also process OIL ontologies to get maximum information within their limitations.

Standard OIL is the complete OIL model, using more primitives than the ones defined in RDF(S). Instance OIL adds instances of concepts and roles to the previous model and Heavy OIL is the layer for future extensions of OIL.

When describing ontologies in OIL we have to distinguish three different layers object level (Instance OIL) where concrete instances of ontology are described. First Meta level (Standard OIL) provides the definitions of ontologies. Second Meta level (Core OIL) describes features of ontology such as author, name, subject etc.

TheFirst Meta level is called ontology definition and later is called ontology container.

OIL ontology is a hierarchy of classes (concepts) with slots (relations) between them. A concept can be anything about which something is said, and, therefore, could also be the description of a task, function, action, strategy, reasoning process, etc.

It possible to define partitions (sets of disjoint concepts). This is an important feature, especially for agents that reason with the information in the ontology. They won't allow an instance to be an instance of two concepts that belongs to a partition.

For example, concepts Table and Chair can define a partition in furniture ontology. We can declare slots (relations between classes) together with logical axioms, stating whether they are functional (having at most one value), transitive, or symmetric, and stating which (if any) slots are inverse. We can state range restrictions as part of a slot declaration as well as the number of distinct values that a slot may have. We can further restrict slots by value-type or has-value restrictions. A value-type restriction demands that every value of the property must be of the stated type; has-value restrictions require the slot to have at least values from the stated type.

4.4.1 Merits:

- OIL provides more expressiveness and complexity than is needed.
- Applications that can only process a lower level of complexity can still catch some of ontology's aspects which are written in OIL.
- An application of higher level of complexity developed in OIL can still understand ontologies expressed in a simpler ontology languages.
- It provides most of the modelling primitives commonly used in frame-based Ontologies.
- It has a simple, clean, and well defined semantics based on Description Logic.

4.4.2 Limitations:

- Inherited valuesfrom super classes cannot be overwritten. As a result OIL lacks in modelling default values.
- Algebraic properties of slots are limited in OIL. There is no facility for describing arbitrary axioms that must hold for the items in the ontology.
- It does not support concrete domains (e.g., integers, strings, etc.).

Table2 4.4:Property Descriptions of OIL

Property Name	**Domain**	**Range**
oil:subClassOf	rdfs:Class	oil:ClassExpression
oil:domain	rdf:Property	oil:ClassExpression
oil:range	rdf:Property	oil:ClassExpression
oil:hasOperand	oil:BooleanExpression	oil:Expression
oil:individual	oil:OneOf	rdfs:Resource
oil:hasPropertyRestriction	rdfs:Class	oil:PropertyRestriction
oil:onProperty	oil:PropertyRestriction	rdf:Property
oil:toClass	oil:PropertyRestriction	oil:ClassExpression
oil:toConcreteType	oil:PropertyRestriction	oil:ConcreteTypeExpression
oil:stringValue	oil:ConcreteTypeExpression	oil:String
oil:integerValue	oil:ConcreteTypeExpression	oil:Integer
oil:individualFiller	oil:HasFiller	rdfs:Resource
oil:stringFiller	oil:HasFiller	oil:String
oil:integerFiller	oil:HasFiller	oil:Integer
oil:number	oil:CardinalityRestriction	oil:Integer
oil:inverseRelationOf	rdf:Property	rdf:Property
oil:hasObject	oil:Axiom	oil:ClassExpression
oil:hasSubject	oil:Covering	oil:ClassExpress

 ISSN: 2321-8169
4047 - 4055

		ion
oil:isCovered By	oil:Covering	oil:ClassExpression

4.5 DAML+OIL (DARPA Agent Markup Language + Ontology Interchange Language)

DAML+OIL [3] [7] has been developed by a joint committee from the US and the European Union (IST) in the context of DAML, a DARPA project for allowing semantic interoperability in XML. It shares the same objective as OIL and builds on RDF(S). In DAML+OIL concepts are called classes it allows sets of disjoint concepts. i.e. each and every concept have a unique object, no two concepts share a single object. It allows concept attributes such as Instance, Class, Local and Global.

These attributes have predefined facets such as Default slot (used to assign value to attribute) value, Type, Cardinality constraints. Taxonomies are used to organize ontological knowledge using generalization and specialization relationships through which simple and multiple inheritance could be applied.

DAML+OIL support taxonomies,

Subclassofwhich specialises generalised concept to specialised concepts.

Superclassofis inverse of ***Subclass of***.

Partitionis a set of disjoint classes.

Table3 4.5:Property Descriptions of OIL

Property name	**domain**	Range
daml:intersectionOf	daml:Class	daml:List
daml:unionOf	daml:Class	daml:List
daml:complementOf	daml:Class	daml:Class
daml:one of	daml:Class	daml:List
daml:onProperty	daml:Restriction	rdf:Property
daml:toClass	daml:Restriction	daml:Class
daml:hasValue	daml:Restriction	*not specified*
daml:hasClass	daml:Restriction	xsd:nonNegativeInteger
daml:minCardinality	daml:Restriction	xsd:nonNegativeInteger
daml:maxCardinality	daml:Restriction	xsd:nonNegativeInteger
daml:cardinality	daml:Restriction	xsd:nonNegativeInteger
daml:hasClassQ	daml:Restriction	rdfs:Class
daml:minCardinalityQ	daml:Restriction	xsd:nonNegativeInteger
daml:maxCardinalityQ	daml:Restriction	xsd:nonNegativeInteger
daml:CardinalityQ	daml:Restriction	xsd:nonNegativeInteger
daml:inverseOf	daml:ObjectProperty	daml:ObjectProperty
daml:equivalentTo	*not specified*	*not specified*
daml:sameClassAs	daml:Class	daml:Class
daml:samePropertyAs	rdf:Property	rdf:Property
daml:sameIndividualAs	daml:Thing	daml:Thing
daml:differentIndividualFrom	daml:Thing	daml:Thing
daml:disjointWith	daml:Class	daml:Class
daml:disjointUnionOf	daml:Class	daml:List
daml:versionInfo	*not specified*	*not specified*
daml:imports	*not specified*	*not specified*
daml:first	daml:List	*not specified*
daml:rest	daml:List	daml:List
daml:item	daml:List	*not specified*

Disjoint decompositions all the properties of a common concept do not necessarily be reflected in all classes of the partition.

Exhaustive decomposition all the properties of a common concept will necessarily be reflected in all classes of the partition. ***Not subclass of***a denial of ***Subclass of***.

4.6 OWL (Ontology Web Language)

Owl [1] [3] [4] [7] [8] is a language used for processing web information. Owl is most popular language in creating ontologies. It is built on rdfs and adds new constructs for more expressiveness to eliminate the weakness in Rdf/s and DAML+OIL. In owl the root class is owl:Thing. owl offers more expressiveness in defining classes. It provides richer integration and interoperability.

The structure of class hierarchy in owl is explained below using diagram.

W3C's classify OWL into three sublanguages, each of which is intended to supply different aspects of these incompatibilities. OWL's sub-languages are OWL Full, OWL Lite, and OWL DL.

4.6.1Namespaces

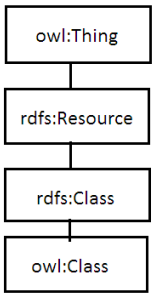
Because OWL is written in RDF, and RDF is written in XML, so OWL documents start with several namespace declarations using RDF, XML Namespace, andURIs. rdf:RDF

is the root element of a OWL Ontology used to specify a number of namespaces.Hence it is called owl Header.These elements may be comments,version specifying statements,uiltin classes or resources imported from other ontologies and labels.

In RDF schema, the root class of everything is rdfs:resource. More specifically, this root class has the following URI: http://www.w3.org/2001/01/rdf-schema#resource

In the world of OWL, the owl:Thing class is the root of all classes; its URI is as follows,http://www.w3.org/2002/07/owl#Thing.

The class hierarchy structure in owl is



4.6.2Owl Document:

Every element in ontology should be specified in between <owl:ontology> and</owl:ontology> elements.Hence it is called OWL ontology.

4.6.2.1 Classes:

owl includes set operators and it define them as classes.Those are

1.owl:unionOf
2.owl:intersectionOf
3.owl:complementOf

4.6.2.2 Enumeration:

Enumeration is another brand-new feature that has been added by OWL.It used to enumerate a large number of instancesto a single one.

This datatype format makes use of the owl:oneOf construct, that is also used for describing an enumerated class. In the case of an enumerated datatype, the subject of owl:oneOf is a blank node of class owl:DataRange and the object is a list of literals. Unfortunately, we cannot use the rdf:parseType="Collection" idiom for specifying the literal list, because RDF requires the collection to be a list of RDF node elements. Therefore we have to specify the list of data values with the basic list constructs rdf:first, rdf:rest and rdf:nil.

4.6.3 Properties:

OWL has some additional properties inaddition to properties from RDF and RDF(S).Those are

- **owl:allValuesFrom:**An object satisfies all properties of a class to become an instance member of that class.For specifying use allValuesFrom to indicate all properties of that class.
- **owl:someValuesFrom and owl:hasValue:**In some cases,An object satisfies only properties of a class to become an instance member of that class.For specifying use someValuesFrom to indicate some properties of that class and those properties are defined using hasValue.
- **owl:cardinality:**using cardinality ,one can specify the restrictions on a class.
- **owl:minCardinality:**It is used to specify the minimum number of restrictions satisfied by class i.e.at least number of properties should be satisfied to become instance member of that class.
- **owl:maxCardinality:**It is used to specify the maximum number of restrictions satisfied by class i.e.atmost number of properties should be satisfied to become instance member of that class.Using both owl:minCardinality and owl:maxCardinality use can specify range.
- **owl:objectProperty:**It is used to specify how one object is related to another i.e. relationship between objects.It is a subclass of rdf:Property.Symmetric and transitive properties are subclasses of object property
- **owl:datatypeProperty:** It is used to connect a resource to a datatype,it may be xmlschemabuilt indata types or literals defined by rdf also.

OWL provides several features to its properties. Those are,

- **SymmetricProperty:**A symmetric property describes the situation in which, if resource R1 is connected to resource R2 by property P, then resource R2 is also connected to resource R1 by the same property.
- **TransitiveProperty:**A transitive property describes the situation in which, if a resource R1 is connected to resource R2 by property P, and resource R2 is connected to resource R3 by the same property, then resource R1 is also connected to resource R3 by property P.
- **FunctionalProperty:** A functional property describes the situation in which, for any given instance, there is at most one value for that property. In other words, it defines a many-to-one.
- **InverseProperty:**An inverse property describes the situation in which, if a resource R1 is connected to resource R2 by property P, then the inverse property of P will connect resource R2 to resource R1.
- **InverseFunctionalProperty:** An inverse functional property, as its name suggests, is just the

 ISSN: 2321-8169
4047 - 4055

opposite of functional property .i.e.for any given instance there must be a unique value.

Table4 4.6:Property Descriptions of OWL

Property name	Domain	Range
owl:intersectionOf	owl:Class	rdf:List
owl:unionOf	owl:Class	rdf:List
owl:complementOf	owl:Class	owl:Class
owl:one of	owl:Class	owl:List
owl:onProperty	owl:Restriction	rdf:Property
owl:allValuesFrom	owl:Restriction	rdfs:class
owl:hasValue	owl:Restriction	*not specified*
owl:someValuesFrom	owl:Restriction	rdfs:class
owl:minCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite:{0,1} OWL DL/Full:{0..N}
owl:maxCardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite:{0,1} OWL DL/Full:{0..N}
owl:cardinality	owl:Restriction	xsd:nonNegativeInteger OWL Lite:{0,1} OWL DL/Full:{0..N}
owl:inverseOf	owl:ObjectProperty	owl:ObjectProperty
owl:equivalentClass	owl:class	owl:class
owl:equivalentProperty	rdf:Property	rdf:property
owl:disjointWith	owl:Class	owl:Class
owl:differentFrom	owl:Thing	owl:Thing
owl:sameIndividualAs	owl:Thing	owl:Thing
owl:distinctMembers	owl:AllDifferent	rdf:List
owl:versionInfo	*not specified*	*not specified*
owl:imports	owl:Ontology	owl:Ontology
owl:priorVersion	owl:Ontology	owl:Ontology
owl:incomaptibleWith	owl:Ontology	owl:Ontology
owl:backwarComapatibleWith	owl:Ontology	owl:Ontology

Merits:

- It provides property chains.
- It provides richer datatypes and data ranges when compared with other semantic languages
- Qualified cardinality restrictions are allowed
- Asymmetric, Reflexive and disjoint are newly added properties in OWL.
- It has more inference power.

Limitations:

- It only allows importing of an entire ontology, not parts of it.
- It does not support overridden so it treats inherited values as default values.
- OWL does not allow the composition of properties for reasons of decidability.
- It does not directly support primitive data type classes.

5. Conclusion:

Ontology languages are the important aspect in the creation of ontology which makes domain knowledge machine readable. These ontologies lead to build semantic web. Thus a brief survey of ontology languages are provided, gives the basic understanding of ontologies, description logic and of frames. Choosing ontology language is the essential step in creating ontology where each language has varying expressive power. In other way ontology languages are chosen according to the ontological knowledge based application needs.

6. References:

[1] OWL Web Ontology Language Reference,http://www.w3.org/TR/owl-ref/#DataRange-defW3C Recommendation 10 February 2004

[2] Jeff Heflin and James Hendler.Dynamic Ontologies on the Web, Copyright © 2000, American Association for Artificial Intelligence (www.aaai.org).

[3] Asunción Gómez-Pérez and Oscar Corcho.Ontology Languages for the Semantic Web, 1094 7167/02/$17.00 © 2002 IEEE INTELLIGENT SYSTEMS

[4] Liyang Yu. Introduction to the Semantic Web and Semantic Web Services, © 2007 by Taylor & Francis Group, LLC Chapman & Hall/CRC is an imprint of Taylor & Francis Group, an Informa business

[5] Oscar Corcho, Asunción Gómez –Pérez. A Roadmap to Ontology Specification Languages, EKAW00.12th

International Conference on Knowledge Engineering and Knowledge Management.

[6] Dieter Fensel and Frank van Harmelen, Ian Horrocks, Deborah L. McGuinness, Peter F. Patel-Schneider. OIL: An Ontology Infrastructure for the Semantic Web, 1094-7167/01/$10.00 © 2001 IEEE INTELLIGENT SYSTEMS.

[7] Mohammad Mustafa Taye. The State of the Art: Ontology Web Based Languages: XML Based Journal of Computing, Volume 2, Issue 6, June 2010, ISSN 2151-9617.

[8] Asunción Gómez –Pérez (asun@fi.upm.es).OntologicalEngineering. ISWC 2008

[9] Shaik.Nazeer,"Benefitsand Barriers of adopting Cloud Computing" ,Internation journal of modern sciences & Engineering Technology,Vol 1 Issue 6,Nov. 2014,pp.117.123.

[10] Shaik.Nazeer,"A Riskless Prudent Data Transfer in Clouds" , International Journal on Recent and Innovation Trends in Computing and Communication , Vol:2,Issue: 9 ,pp no.2780-2785.,Sept.2014.

[11] Shaik.Nazeer, "Security Constraints in Cloud Computations", International Journal of advanced computer, electrical and electronics engineering, Vol. 1, No. 1, 75-80, Oct., 2012.

[12] Shaik.Nazeer, "Intrusion Dection using Knowledge Dicsovery Method", CIIT international journal of Data mining and Knowledge Vol 4,No.5,,May 2012pp.226..232.

[13] Shaik.Nazeer, "A Study on the Readiness of Cloud Computing for Captious Computations" ,International Journal World of Computer Science and Information Technology (WCSIT), Vol. 1, No. 6, 247-252, Aug., 2011.

[14] Shaik.Nazeer, "Raw Era in Cloud Computing", CIIT international journal of Networking and Communication Engineering,Vol 3, No.13,October 2011,pp 833 to239.

[15] Shaik.Nazeer, "Semantic Web Security and Privacy", Journal of Theoretical and Applied Information Technology (JATIT), Vol.22, Issue.1, pp.9-18, Dec., 2010.

R.Pavan Kartheek is Asst.Prof in Computer science and Engineering at Bapatla EngineeringCollege.His research intrests include Semantic Web and Word Sense Disambiguation. He received M.Tech in Computer Science and Engineering from Vignan University.
Email:pavankartheek.r@gmail.com

M.Gamya is pursuing B.Tech in Computer Science and Engineering at Bapatla Engineering College. Her research interests include Semantic Web and Networks. Email: gamyamurikipudi@outlook.com

Dr. Shaik Nazeeris an Associate Professor in Dept.of Computer Science& Engineering at Bapatla Engineering College, Bapatla. He has 12years of Teaching Experience and 2 years of Industry Experience. He had published more than25Papersin international Journals/ Conferences. As a Co-inventor he got 3 utility patents from State Intellectual Property Office, China. He is associated to many professional bodies like ISTE, CSI, IDES, IAENG, SDIWC, IACSIT; His areas of interest are Network security, Cloud Computing,Big Data Analytics.

Email: shk_nazir@yahoo.co.in