4 Ontology

Objective:

This chapter explains in detail, the role of ontology in building Web 3.0 and the advantage of ontology over traditional hierarchical structures. The chapter also discusses, in short, about protégé which is a tool to generate ontology. The complete procedure to generate ontology using protégé is explained in detail in chapter 6.

4.1 Introduction to Ontology

Going by the traditional definition, 'Ontology can be defined as a branch of metaphysics concerned with the nature and relations of being.' Some questions always get related to ontology, whenever an attempt is made to infer the relation of beings. Stated below are the principal questions involved:

- "What can be said to exist?"
- "Into what categories, if any, can we sort existing things?"
- "What is the meaning of being an entity?"
- "What are the various modes of being an entity?"

We can use the same strategy to build ontology in Semantic Web, i.e. You must make sure that the ontology which you have built answers the above stated fundamental questions. This will help you to conclude that you have included all the essential elements required by your machine to understand the fact that you are trying to put forward. The above mentioned statement's logic will be more meaningful, once you completely understand the concept of ontology.

Definition of Ontology

Ontology is an explicit and abstract modeled representation of already defined finite sets of terms and concepts, involved in knowledge engineering, knowledge management and intelligent information integration. To be more specific, I can define Ontology as an 'explicit specification of conceptualization' (stated by Thomas Gruber).While the terms specification and conceptualization have caused much debate, the essential points of this definition of ontology are:

- Ontology defines (specifies) the concepts, relationships, and other distinctions that are relevant for modeling a domain.
- The specification takes the form of the definitions of representational vocabulary (classes, relations, and so forth), which provide meanings for the vocabulary and formal constraints on its coherent use.

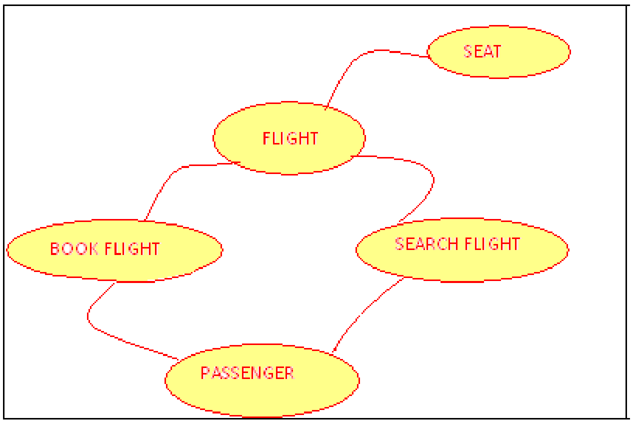

Figure 4.1: Sample Ontology

Conceptualization

Conceptualization contains the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold them. A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose.

A conceptualization can be defined as a tuple (U, R) where,

- U is a set called as a universe of discourse.
- R is a set of relations on U.

A conceptualization is an abstract, simplified view of the world that we wish to represent for some purpose. Every knowledge base, knowledge-based system, or knowledge-level agent is committed to some conceptualization, explicitly or implicitly.

Note:
In the context of ontology, **formal** means machine understandable. And **share** means consensual knowledge accepted by a group.

4.1.3 Scope of Ontology

Ontology defines a common vocabulary for researchers who need to share information in a domain. It includes machine-interpretable definitions of basic concepts in the domain and the relationship among them. Ontology is a description (like a formal specification of a program) of the concepts and relationships that can exist for an agent or a community of agents. In general, the subject of ontology is the study of the categories of things that exist or may exist in some domain. The product of such a study is a catalog of the types of things that are assumed to exist in a domain of interest D from the perspective of a person who uses a language L for the purpose of talking about D. The types in the ontology represent the predicates, word senses, or concept and relation types of the language L when used to discuss topics in the domain D. Logic such as predicate calculus, conceptual graphs, ontology or KIF that is not interpreted, is ontologically neutral. It imposes no constraints on the subject matter or the way the subject may be characterized.

As an interface specification, the ontology provides a language for communicating with the agent. An agent supporting this interface is not required to use the terms of the ontology as an internal encoding of its knowledge. Nonetheless, the definitions and formal constraints of the ontology do put restrictions on what can be meaningfully stated in this language. In essence, committing to ontology (e.g. supporting an interface using the ontology's vocabulary) requires that statements which are asserted on inputs and outputs are logically consistent with the definitions and constraints of the ontology. This is analogous to the requirement that the rows of a database table (or insert statements in SQL) must be consistent with integrity constraints, which are stated declaratively and independently of internal data formats.

Similarly, while ontology must be formulated in some representation language, it is intended to be a semantic level specification, i.e. it is independent of data modeling strategy or implementation. For instance, a conventional database model may represent the identity of individuals using a primary key that assigns a unique identifier to each individual. However, the primary key identifier is an artifact of the modeling process and does not denote something in the domain. Ontologies are typically formulated in languages which are closer in expressive power to logical formalisms such as the predicate calculus. This allows the ontology designer to be able to state semantic constraints without forcing a particular encoding strategy. Similarly, in an ontology one might represent constraints that hold across relations in a simple declaration (A is a subclass of B), which might be encoded as a join on foreign keys in the relational model.

4.2 Switching from database to Ontology

Owing to technological determinism, we are always focused on the next glittering innovation. The one standing in the forefront of this innovation queue is ontology. Being students of computer engineering, developing the front-end and back-end of a web application is no more a big deal, but a challenge is certainly faced when it comes to making the system more intelligent. Under such circumstances, the best way would be to migrate from database to ontology. Ever since the introduction of web 1.0 which is actually a read-only platform for information, to the Web 2.0 which is supposed to be a platform for participation, emphasis has always been laid on developing a more nuanced way to organize information. Basically, ontologies work to organize information. No matter what the domain or scope is, ontology is a description of a world view using a linked or networked graph structure. Taking a little bit of diversion, let's have a look into a more common term, i.e. 'Relational database management system (RDBMS)'. We have many systems that are based on RDBMS. In fact, most of the current folksonomy use RDBMS as their base. The most obvious reason behind this is that database management system is a standard way to store data on a permanent basis and that the extraction of data can be easily done using SQL.

But consider a case wherein you have several databases represented in various formats and your application insists on an integration of these databases, you will face several problems because of their different formats. This is the area where ontology gains weightage. By virtue of the relationship structure underlying ontology, they are excellent vehicles for discovery and linkages. Parsing through this relationship graph is the basis of the Concept Explorer. Separating domain knowledge from operational knowledge and enabling their reuse, sharing a common understanding of the structure of information among software agents are some of the important goals implemented through the medium of ontology. Like for instance, if there are several different web sites containing information about medicines and if these Web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications. Thus, they provide knowledge sharing and reuse among both human and computer agents because of their ability to interweave human and machine understanding through formal and real-world semantics.

4.2.1 Taxonomy as a pre-cursor of Ontology

Learning a concept in Semantic Web is not an easy task. This is because, most of the topics are research-oriented and in order to properly understand the definitions of these concepts, an in depth understanding of the terms that construct the definition is a must. The easiest way to do this is by trying to relate these technical terms with their corresponding dictionary meaning. The word taxonomy is derived from two Greek words- Taxis and Nomos which mean 'the arrangement and ordering of things' and 'anything assigned, usage or custom, law or ordinance' respectively. In a formal way, taxonomy can be defined as a subject-based classification that arranges the term in a controlled vocabulary and allows related terms to be grouped together and categorized in ways that make it easier to find the correct term to use.

Many taxonomies have a hierarchical structure, but this is not a requirement. Taxonomies can be explained in simple terms as a graphical representation of classification of things, ideas, etc. According to some taxonomic scheme almost anything can be classified, as long as they have a logical hierarchy. They work towards organizing information. The backbone of ontology is often taxonomy. Taxonomy is a classification of things in a hierarchical form. It is usually a tree or a lattice that expresses the subsumption relation. An example is classification of living organisms. The taxonomy usually restricts the intended usage of classes, where classes are subsets of the set of all possible individuals in the domain.

Example of taxonomy:

When we try to compare the concept of taxonomy and ontology we will find that both of them run on similar concepts. For layman or end-users it's almost the same concept. But technocrats are obliged to perceive them separately. This is because ontology implies a broader perspective of taxonomy. Taxonomy is a mere classification of objects for better understanding of objects and their sub-types, whereas ontology represents a complex network of interconnections between the nodes that focuses mainly on a structure that enables inference and logical conclusions.

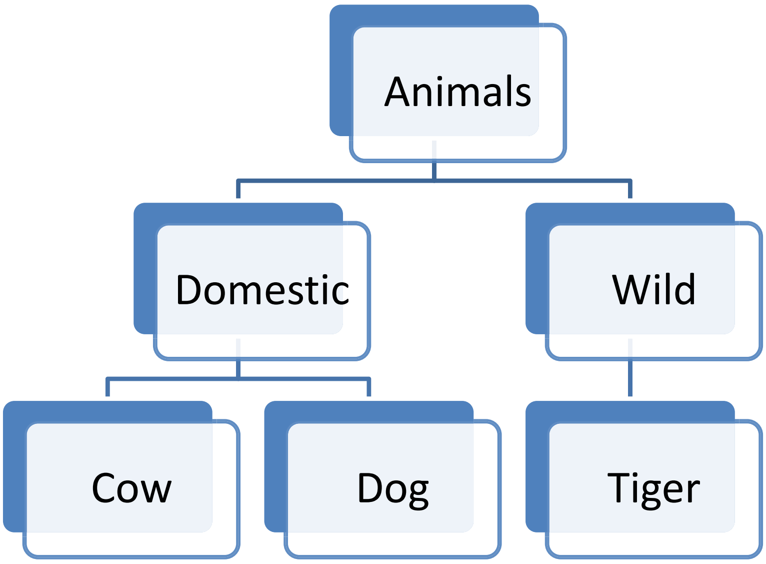

Figure 4.2: Graphical representation of taxonomy.

4.3 Difference between Ontology and taxonomy

Ontology	Taxonomy
Ontologies can be defined as an explicit specification of conceptualization.	Taxonomy is defined as a classification of elements in an ordered system that indicates natural relationships.
Ontologies generally include descriptive terms, in order to specify the vocabulary of the node contents.	Taxonomies generally do not include descriptive keywords for each item.
Ontologies represent a very complex form of interconnection.	Taxonomies are comparatively simpler and thrive to organize a subject in a particular way.
Relationships in ontology adhere to subjective interpretation, thus, triggering inference.	Taxonomies tend to be quite lenient about the kind of relationship that exists between parents and child node in the classification tree.

4.4 Types of Ontology

Several types of classifications are proposed for ontologies based on the characteristic of ontology's components. Explained below are few of the important types of classification, proposed for ontology.

4.4.1 Classification according to purpose

Application ontology

Application ontology is used in specific applications, in which a reasoner is implemented based on ontology.

Reference ontology

Reference ontology is used during development of applications, for mutual understanding and explanation between agents belonging to different communities, for establishing consensus in a community that has to adopt a new term or simplify a term for explaining the meaning of it to somebody new to the community.

4.4.2 Classification according to expressiveness

Heavyweight ontology

Heavyweight ontologies are heavily axiomatized and thus, represent an ontological commitment explicitly. The purpose of the axiomatization is to exclude terminological and conceptual ambiguities due to unintended interpretations.

Lightweight ontology

Lightweight ontologies are simple taxonomic structures of primitive or composite terms together with associated definitions. They are hardly axiomatized as the intended meaning of the terms used by the community is more or less known in advance by all members and the ontology can be limited to those structural relationships among the terms that are considered as relevant.

4.4.3 Classification according to specificity

Generic ontology

The concepts defined by this layer are considered to be generic across many fields. Typically, generic ontologies define concepts such as state, event, process etc.

Core ontology

Core ontologies define concepts which are generic across a set of domains. Therefore, they are situated in between the two extremes of generic and domain ontologies. The borderline between generic and core ontologies is not clearly defined because there is no exhaustive enumeration of fields and their conceptualizations. However, the distinction is intuitively meaningful and useful for building libraries.

Domain ontology

Domain ontologies express conceptualizations that are specific for a specific universe of discourse. The concepts in domain ontologies are often defined as a specialization of concepts in the generic and core ontologies. The borderline between core and domain ontologies is not clearly defined because core ontologies intend to be generic within a domain. Thus, it is usually hard to make a clear cut between generic and core as well as between core and domain ontologies. A concept such as a software component would be placed in core ontology for application servers.

4.5 Why to develop Ontology?

Some of the reasons to develop ontology are:

- To share a common understanding of the structure of information among people or software agents.
- To enable reuse of domain knowledge.
- To make domain assumptions explicit.
- To separate domain knowledge from the operational knowledge.
- To analyze domain knowledge.

Sharing common understanding of the structure of information among people or software agents:

This is one of the most common goals in developing ontologies (Musen 1992; Gruber 1993). For example, suppose that several different web sites contain medical information or provide medical e-commerce services. If these web sites share and publish the same underlying ontology of the terms they all use, then computer agents can extract and aggregate information from these different sites. The agents can use this aggregated information to answer user queries or as input data to other applications.

Enabling reuse of domain knowledge:

This was one of the driving forces behind the recent surge in ontology research. For example, models for many different domains must represent the notion of time. This representation includes the notions of time intervals, points in time, relative measures of time, and so on. If one group of researchers develops ontology in detail, others can simply reuse it for their domains. Additionally, if we need to build a large ontology, we can integrate several existing ontologies describing portions of the large domain. We can also reuse a general ontology, such as the UNSPSC ontology, and extend it to describe our domain of interest.

Making explicit domain assumptions

Explicit domain assumptions, underlying an implementation, make it possible to change the assumptions easily, if our knowledge about the domain changes. Hard-coding the assumptions about the world in programming-language code, makes these assumptions not only hard to find and understand, but also hard to change, in particular for someone without programming expertise. In addition, explicit specification of domain knowledge is useful for new users who must learn the meaning of terms in the domain.

Separating the domain knowledge from the operational knowledge

We can describe a task of configuring a product from its components, according to a required specification and implement a program that does this configuration independent of the products and components themselves (McGuinness and Wright 1998). We can then develop ontology of PC-components and characteristics and apply the algorithm to configure made-to-order PCs. We can also use the same algorithm to configure elevators if we feed elevator component ontology to it (Rothenfluh et al. 1996).

Analyzing domain knowledge

Analyzing domain knowledge is possible once a declarative specification of the terms is available. Formal analysis of terms is extremely valuable while attempting to reuse existing ontologies and while extending them (McGuinness et al. 2000). Often ontology of the domain is not a goal by itself. Developing ontology is akin to defining a set of data and their structure, for other programs to use. Problem-solving methods, domain-independent applications, and software agents use ontologies and knowledge bases built using ontologies as data.

4.6 Ontology development life-cycle

While building an ontology, the following questions will be helpful, in order to understand where to start and stop the ontology development process.

- What are the activities involved in the ontology development process?
- What is the goal of each activity?
- When should I carry out each activity?
- What is the relationship of one activity with the others?
- Where can I find ontologies with the goal of reusing them?
- How can I build the ontology for my application?
- Do I need a single ontology or an ontology network?

The different stages involved in the development life-cycle of ontology are listed below:

Identify the purpose and scope:

Every project, projected to develop using ontology as the base, will have a scope. This can be well-understood by comparing this stage with the requirement-analysis stage in application development. Developing a requirement specification for the ontology by identifying the intended scope and purpose of the ontology is the initial stage of ontology development life-cycle. A well-characterized requirement specification is important to the design, evaluation and re-use of ontology.

Knowledge Acquisition:

Knowledge Acquisition can be defined as the process of acquiring domain knowledge, using which ontology can be built. Different possible scenarios are identified and clubbed together. Informal competency questions are formed using it.

Conceptualization:

Conceptualization can be defined as a process of identifying the key concepts that exist in the domain, their properties and the relationships that hold between them; identifying natural language terms to refer to such concepts, relations and attributes, and structuring domain knowledge into explicit conceptual models. This is the process touched upon, in the beginning of this chapter, where the concepts and relationships describing the domain were captured. The ontology is usually described using some informal terminology. Gruber suggests writing down the lists of the concepts to be included in the ontology and exploring other ontologies to re-use all or part of their conceptualizations and terminologies.

Encoding:

Encoding is the process of representing the conceptualization in some formal language, e.g. frames, object models or logic. This includes the creation of formal competency questions in terms of the terminological specification language chosen (usually first order logic).

Integrating:

Integrating means to use or specialize in an existing ontology. This task is frequently hindered by the inadequate documentation of existing ontologies, notably their implicit assumptions. Using a generic ontology, gives a deeper definition of the concepts in the chosen domain.

Documentation:

Documentation includes informal and formal complete definitions, assumptions and examples that are essential to promote the appropriate use and re-use of ontology. Documentation is important for defining the exact meaning of terms within the ontology.

Evaluation:

Evaluation is determining the appropriateness of ontology for its intended application. Evaluation is done pragmatically, by assessing the competency of the ontology to satisfy the requirements of its application, including determining the consistency, completeness and conciseness of an ontology. Conciseness implies an absence of redundancy in the definitions of ontology and an appropriate granularity.

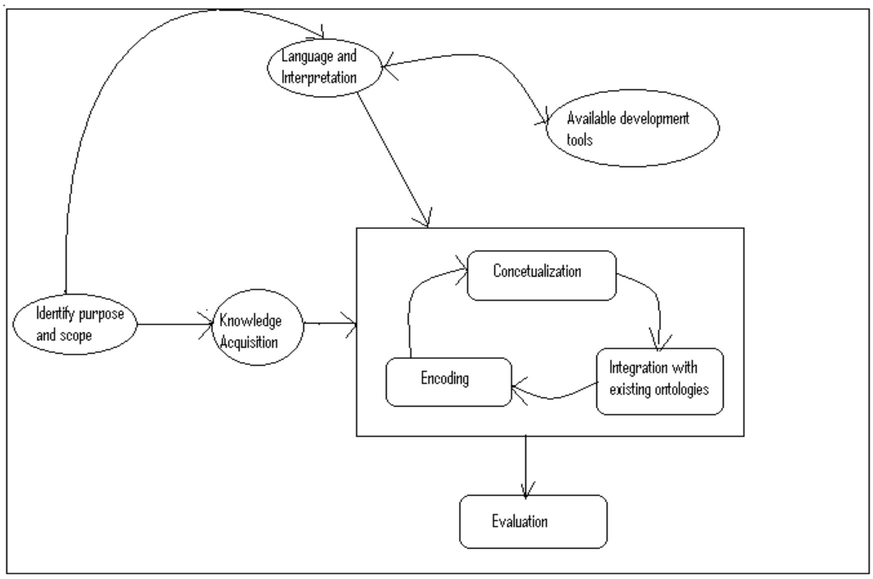

Figure 4.3: Ontology development life-cycle

4.7 Ontology Usage

The following application-independent lifecycle activities are performed at runtime to support ontology:

Ontology Population:

To populate the knowledge base (KB), instances may be collected from the user, e.g. via forms. A substantial overhead may be imposed on the user when all instances of data have to be created manually. This burden can be alleviated by a (semi)-automatic population of the KB. A part of this knowledge-creation step, also involves the manipulation and deletion of instances.

Cleansing and Fusion:

Automatically extracted knowledge cannot be assumed to have the desired quality. Enhancing the instance of data may include identification and merging of conceptually identical instances that are differently labeled (object identification).

Ontological Commitments

In order to share the knowledge amongst the agents, an agreement must exist on the topics which are being communicated. This raises the issue of ontological commitment. Ontological commitments allow a number of agents to meaningfully communicate about a domain without necessarily operating on a globally shared theory. In the context of multiple agents, a common ontology serves as a knowledge-level specification of the ontological commitments of a set of participating agents. A common ontology defines the vocabulary with which queries and assertions are exchanged among the agents, thereby providing the means to bridge the semantic gap that exists between the lexical representations of information and its non-lexical conceptualization.

4.7.1 Ontology Learning

Ontology learning is defined as the set of methods and techniques used for building ontology from scratch, enriching, or adapting an existing ontology in a semi-automatic fashion using distributed and heterogeneous knowledge and information sources, allowing reduction in time and effort needed in the ontology development process. Though a fully automatic acquisition of knowledge remains inaccessible, the overall process is considered as semi-automatic, i.e. human intervention is necessary in some parts of the learning process. New concepts are identified using natural language analysis techniques over the resources previously identified by the user. The resulting ontology is pruned and then focused on a specific domain by means of several approaches based on statistics. Finally, the relation between the concepts is established by applying learning methods.

4.7.2 Ontology Alignment

Ontology alignment consists of establishing different kinds of mappings (or links) between two ontologies, hence preserving the original ontologies. Ontology merging proposes to generate a unique ontology from the original ontologies. We will assume that a mapping between ontologies means rewriting a set of rules that associates terms and expressions defined in source ontology with terms and expressions of a target ontology.

4.8 Advantages of Ontology

A good ontology offers a composite suite of benefits that are not available in taxonomies, relational database schema, or other standard ways, to structure information. Some of these benefits are:

- **Ontologies promote coherent navigation** by enabling the movement from concept to concept in the ontology structure.
- **They have flexible entry points,** because any specific perspective in the ontology can be traced and related to all of its associated concepts; there is no specific set structure or manner for interacting with the ontology.
- **Connections** highlight related information and aid prompt discovery without requiring prior knowledge of the domain or its terminology.
- **Ability to represent any form of information**, including unstructured (say, documents or text), semi-structured (say, XML or Web pages) and structured (say, conventional databases) data.
- **Inferencing,** whereby by specifying one concept (say, mammals) one knows that we are also referring to a related concept (say, that mammals are a kind of animal).
- **Concept matching**, which means that even though we may describe things somewhat differently, we can still match to the same idea (such as glad or happy both referring to the concept of a pleasant state of mind).
- Thus, this means that we can also **integrate external content** by proper matching and mapping of these concepts.
- **Reasoning**, which is the ability to use the coherence and structure it to inform questions of relatedness or to answer questions; this latter benefit is more related to machine learning or artificial intelligence, and is not generally expressed in simpler, standard ontologies.

4.9 Limitations of Ontology

Though ontology is contributing to the progress of Semantic Web, it also has some limitations.

- Ontology makes the abstract model of a particular domain based on set of data and structures but does not define the boundaries of the model.
- Size of ontology varies with respect to the number of classes and instances; if the number of instances is increased to a large extent then it becomes very hard to manage manually, and currently there is no mechanism to manage it automatically.
- Manual ontology generation process sometimes becomes very complex and time consuming, especially while dealing with a large amount of data.