CHAPTER 5

eLearning Services Using Semantic Web Technologies

- Introduction
- Issues in present eLearning System
- System Perspectives
- Ontology Characteristics
- Ontology and eLearning
- Conceptual Framework of Semantic eLearning
- System Architecture
- Ontology Framework
- Learning Design
- Conclusion
- References
5.1 Introduction
The key characteristic of Information and Communication Technology (ICT) is its ability to provide flexible access to information and resources. Flexible access refers to access and use of information and resources at a time, place and pace that are suitable and convenient to individual learners. By ICT, educators are presented with what is known as eLearning, where education is delivered online on a mass scale. The goal was to free individual learners from the constraints of traditional residential educational systems where one had to physically attend lectures at a stipulated time.

Most of today’s eLearning systems provide static learning materials that are based upon the one-size-fits-all philosophy [1]. Essentially they are incapable of generating learning materials dynamically based upon the learners’ requirements. For example, they cannot execute a complex query which can take care of student learning style, student background knowledge, availability of student network connectivity and so on. Like any other information retrieval system, eLearning system in particular, faces challenges when it comes to contextual information retrieval. Some open issues plaguing present eLearning systems are discussed in this chapter.

Depending upon those issues involved in the present eLearning system; the learners' characteristics (discussed in chapter 2) and considering the other factors, a system competency list is presented here. These are mainly from two different perspectives, such as, student as well as tutor and/ or administrator perspectives.

Ontology is considered as the backbone of our system. We believe that many of the issues related to the information retrieval of the present eLearning system can be solved by enabling the ontology-supported system. And this is not only applicable for eLearning system, but it is going to play an important role in the 21st century online information retrieval systems in general. This chapter discusses about ontology and its characteristics, and
some instances of how ontology is going to play a role in eLearning scenario.

In [2] the authors mentioned about the three basic concepts, such as, content, context and structure, which need to be considered while developing eLearning system. Based on these, a conceptual framework of semantic eLearning is shown here, which we name as semantic learning layer cake. Based upon this conceptual framework, we design our system architecture as shown in section 5.7. It is a four layered architecture, from bottom to top viz., knowledge base, inference engine, service layer and the user interface. Then we discuss the ontological frameworks which are the basis of the different ontologies, such as, domain ontology, document ontology and student ontology used as the backend of our system. The application profiles for domain ontology, document ontology and student ontology are provided in Appendix – A, B and C respectively. We also discuss the services. A few use cases are also demonstrated.

5.2 Issues in Present eLearning System
There are many issues involved with the present eLearning systems. They are multi-faceted, for example, from students’ perspectives, tutors perspectives, system administrator perspectives, and so on. We mainly discuss here the core issues in general [2, 3, 16, 27]. It is worth to mention here that most of the following issues came out while discussing with the colleagues in Canada.

1. Lack of group and personalized learning spaces- at present all most all the eLearning systems are based upon the “one-size-fits-all” philosophy. That means they are static in nature and make available the same learning resources for everyone without considering the ability, capacity or intended use of the learning resources by the learners. There is a serious lack in providing the learning resources either customized/ or group based or personalized services.
2. **Presentation of the entire learning material instead of relevant information actually sought by a learner**- this follows the first issue. It is found that even when the intention is to use only a piece of information from a large chunk of data, but still large chunk of data is served.

3. **Learning sequence**- there is also a serious lack in providing the flexible learning sequence, though it is important in a self-learning space. For a learner it always difficult to understand which learning object is to be taken first and which one is to be taken next. System must secure the learning sequence for each individual learners depending upon their characteristics.

4. **Reusability**- it is another important aspect in context of eLearning. While an education establishment develops a learning resource (for example, a diagram, a learning text), it spends lot of resources in terms of monetary, human resources, energy, time and so on. So, it is logical that once a resource is developed it must be reusable as long as it retains its value, instead of regenerating it every time the need arises.

5. **Lack of semantic interoperability**- it is also an important aspect related to sharing the resources. Not only sharing, but also for the third party software agents to access them, compile and to generate the services.

6. **Quality assurance**- this basically points to the reliability of the retrieved learning materials. How can a learner rely on certain learning resources? How does a learner judge the quality of the resources provided by the system? Hence, it is important to have a mechanism to develop the confidence of the learners regarding the quality of the learning resources.

7. **Ranking**- this is also an important thing to consider. How does a learner choose the best quality learning resource from the list of system retrieved learning resources? Which parameter will make a learner ascertain that the chosen material is best suitable for him/her? So, it would be nice if the system ranks the learning materials...
providing also the parameters used to rank them and allow the learners to choose the materials from the list.

The main reasons behind most of the above mentioned issues are the lack in systematic organization and describing the learning resources. In most of the cases, the resources are not enough fine-grained, described and structured. Most of the issues described above, such as, those related to creating personalized and customized/ group based learning space, can be overcome by providing the fine-grained description and structuring them. The fine-grained descriptions of learning objects also allow the semantic interoperability and composition of the learning objects to be carried out on the fly. It is often seen that in order to reuse the learning content of a document (e.g., a paragraph, an illustration, a table, etc.), we copy and paste the content into a new document. It is quite possible to reuse the learning objects in a more sophisticated way if we can access the specific components of a learning object and re-purpose them on-the-fly [3]. Also the resources possess fine-grained description provides a good visualization in devices with small windows [4].

Specifically, we can formalize an eLearning system into three different levels, i.e. content, context and structure, where, content identifies what the learning material is about, context identifies in which form this topic is presented, while structure is to comprehend the set of learning materials in a learning course [2]. In our present work we tried to address the first five enumerated issues as enlisted above, with an objective of achieving:

1. Learner centric educational architecture
2. Interest based knowledge retrieval
3. Achieving reusability
4. Achieving semantic interoperability
5.3 System perspectives
This section discusses some of the factors in the light of semantic web technologies enabled eLearning systems. These factors play very crucial role in developing our system. They also play a crucial role in developing the services. In order to generate services we used these factors. Basically we considered these factors as system competency questions and we translated them into necessary services required by the students. The service encoding is shown in section 5.9. These factors can be seen from two different perspectives, from students’ perspectives and from the tutors and/or administrators perspectives.

Student perspectives
Factor 1: Style of learning
As we discussed in chapter 2, different students build up their own style of learning. For example, some students may prefer to learn by just going through the text, whereas, other may prefer to learn by going through the visual representation of the learning contents and so on. The system must deliver the learning materials accordingly based upon the individual students learning style.

Factor 2: Cognitive style of learning
Cognitive style (i.e., ability to link one thing to another) is another important factor to be considered at the time of designing the self-learning paced eLearning system. Every student has their own way of learning (different cognitive style of learning is discussed in chapter 2). They have their own way of thinking, processing, building and storing knowledge. For example, some student may always prefer to learn by first going through the examples and then taking up the theory. Similarly, some student may prefer to learn the theory first and then prefer to do some exercises to apply what she/ he learned and so on. So here the most significant thing is that the system should provide the materials in a sequence based upon the students’ cognitive style of learning.
Factor 3: Language
Language is an important issue in the self-learning space. It is important that the system must consider student competency/disability towards a particular language. For example, a student competent in English language must always receive the learning materials in English language instead of some other language which she/he cannot comprehend.

Factor 4: Network connectivity
Network connectivity speed is an important issue particularly in respect to the multimedia materials while transferring over a network. Even though it is not an issue at present particularly in context of the developed countries, still it is an important issue to the developing or underdeveloped countries. To access a video material (e.g., lecture videos) we need more bandwidth whereas it is not the case with the materials in text format. So, system should verify what bandwidth maybe needed to access a particular resource and cross verify if the student is indeed connected at that speed. If no, maybe system can make recommendations of the same material in text formats or such alternate items.

Factor 5: Education level
Education level means a specific identifiable position in the education process. For example, the education level can be defined as “primary level”, “secondary level”, “higher level” and so on. It can also be defined as “basic level”, “advanced level” and so forth. So, we can classify the students based upon their level of education. Similarly, this classification is applicable for the learning materials also as we believe that each document has its own specific user. It would not be acceptable to us that a secondary level student and a masters’ level student receive the same learning material. The system must be in a position to distinguish between those two students. So, the system must deliver only those materials, which fit to a student as per his/her education level.
Factor 6: Student background knowledge
Classroom teaching is handled by a teacher. Here, a teacher is aware about students’ previous knowledge. That means a teacher knows what to deliver and what not to and at what sequence. Since eLearning is a self-learning space, it is important that system must recognise student’s knowledge on each part of the learning content already taken and deliver the suitable learning materials accordingly. In fact in this case, a semantic based eLearning system is in a better position than the classroom teaching. In a classroom teaching it is quite difficult for a teacher to facilitate the learning at personalised level whereas the semantic based eLearning system can facilitate it.

Factor 7: Required prerequisite knowledge
There can be a set of prerequisites for students to register with a course or to access the materials on a particular topic. So, the system must be able provide the list of prerequisites knowledge require to register with a course or to access the materials on a topic.

Factor 8: Matching between student background knowledge and the prerequisites
The system must be able to verify the student background knowledge with the prerequisites of the learning materials on a topic before delivering the materials.

Tutor and / or administrator perspectives
Factor 9: Reuse
System should allow the tutors to search, retrieve and reuse the learning resources in creating new lessons.

Factor 10: Track the learners’ progress
System should allow the tutors to track the learner’s progress.
5.4 Ontology Characteristics

According to Gruber, ontology is defined as, “an explicit specification of a conceptualization” [5]. Later on Studer et al [6] extended the definition and defined ontology as "a formal, explicit specification of a shared conceptualization".

Studer’s definition includes the idea of shared in the notion of conceptualization and formal relations among the concepts. The explicit, formal representation of a shared conceptualization involves a perspective of a specific reality, and which is constituted in the conceptual structure of a knowledge base. Furthermore, the ultimate objective of ontology is to share the knowledge it represents. An ontology defines the terms and their formal relations within a given knowledge area.

So, we can define ontology as, a set of shared conceptualizations with their formal relationships developed in a polynomial hierarchy.

Based upon the above discussions, we can draw the characteristics of ontology as follows,

1. **Shared**- the notion of ontology is to capture commonly agreed knowledge.

2. **Conceptualization**- refers to the mental formulation of a phenomenon in the world. It is developed by identifying the related concepts of that phenomenon. The shared conceptualization will help in avoiding the lack of shared understanding between terms in one vocabulary as well as terms in various metadata vocabularies. So, searching no longer need to be content matching based, but would be based upon the true meaning of the concepts.

3. **Formal relationships**- ontology formalizes the relationships among the concepts, which makes computer to interpret the semantic relationships among the concepts and infer the implicit knowledge.
4. *Polynomial hierarchy*- it develops in a polynomial hierarchy instead a rigid monolithic hierarchy structure.

### 5.5 Ontology and eLearning
In order to describe the learning objects, different communities use different metadata standards as per their requirements. We know that metadata elements lack a formal semantics, as they are mainly useful in indexing according to popular access points such as 'creator', 'publisher' etc. So when it comes to sharing resources between heterogeneous domains instead of homogenous domains (also many a time intra-domain), we face the problem of incompatibility. The lack of shared understanding between terms of various metadata vocabularies might be avoided by using ontologies as a conceptual backbone in an eLearning scenario [2]. In eLearning, ontology in general helps us to define the learning components more strictly.

The purpose of ontology is to achieve interoperability by providing a common terminology and understanding of a given domain of interest, which in turn allows for the assignment of a clear meaning to learning materials. Our system is implemented in the semantic web environment, and emphasizes the fact that:

1. Standards are concerned about *semantics* rather than just about *syntax*
2. Extensible methods for data integration should be provided in eLearning environment

### 5.6 Conceptual Framework of Semantic eLearning
In order to deal with the above mentioned problems, we formalize a conceptual learning space and call it a *semantic learning layer cake*. Each layer is built on top of another layer. These layers are formed in a bottom-up approach [figure 5.1].
The bottom layer contains the *content objects*. The set of *content objects* form the *learning object* layer. Figure 5.2 shows the internal structure of the learning objects. We perceive *learning objects* as an aggregation of *content objects* whereas each *content object* is formed by a set of *content fragments* (or content units). The *content fragments* are learning content elements in their most basic form, like text, image (e.g., paintings, graphics, moving images), sound, datasets (e.g., tables, lists), etc. They represent individual resources in isolation. As shown in the figure above that the aggregated content fragments define the navigation within a content object, which helps us in defining the learning path.
In the semantic learning layer, on top of learning object is the *content* and *context*. The *content* contains the concepts or subject terms defining the “thought content” or “semantic content” of the learning objects. In our framework, we formalize content as “domain ontology”. “Domain ontology” helps in reducing the knowledge gap between the teacher and learner by formulating the unambiguous and shared understanding of terms. It also helps to overcome the problem of synonym, homonym, antonym, etc. and other related problems (e.g., acronym) that we often face in an online information retrieval environment. The *context* identifies the facts or circumstances of the learning objects. In our framework, the *context* is represented from three different angles, such as, matching the education level of both the document and the learner, intended use of the learning object and the learning objectives. The details are provided in the following sections.

On top of *content* and *context* is the *structure* layer. *Structure* formalizes the relations between the learning materials. The relations are specified by the properties, such as, *hasPart*, *isPartOf*, *hasPrerequisite* and *isPrerequisiteOf*. These relations help in defining the learning sequence. *Learning design* is the top most layer of the semantic learning cake. This layer uses the students profile and other layers below it to create a personalized learning environment with the aid of sequential activities.
Essentially, it specifies the roles, sequence, logistical information and pedagogical information. The *learning design* layer is formalized using logic rules.

### 5.7 System Architecture

In this section we discuss the system architecture. It is developed in a multi-layered structure. The layers are organized in the following order, from bottom to top, such as, knowledge base, inference engine, service layer and finally on top, is the common integrated user interface. The description of each of the layers is discussed in detail below. We also mention about the tools and technologies used in developing the system in each of the respective layers. The following figure 5.3 shows the system architecture.
Figure 5.3: System architecture

**Knowledge base:** it is the storehouse of the system. It stores the ontologies, such as, domain ontology, document ontology and learning ontology. These ontologies are primarily created in OWL-DL using an ontology editor Protégé [7] and stored in the system knowledge base. The ontology classes and sub-classes of document ontology are created by selecting the metadata elements from IEEE LOM [8] and Dublin Core [9] wherever applicable. We extend the use of these metadata elements into object and datatype properties as well wherever applicable. In addition to use of standard metadata elements in our ontology model, we define more classes, sub-
classes and properties, wherever needed to fulfill architectural and system requirements. However, importance is given to use of standard metadata elements in our model as much as possible, as one of our system goals is to achieve semantic interoperability. So, wherever applicable we tried to keep our self in following the standards, so that our system can easily communicate with the third party software agents on the Web.

The same principle is applied in building the student ontology as well. In student ontology we mainly use IMS Learners Information Package [10] and vCard [11] metadata standards in addition to our own defined classes, sub-classes and properties.

In building domain ontology, we used SKOS specifications and then we extended the same as per our requirements (details are described in the section 5.8.1.3).

Another important part of the knowledge base is a set of rules. These rules act as nerves of the system. These rules are written using N3Logic. The impact of these rules in generating the services, are discussed in detail in section 5.9. The knowledge base also includes the annotated learning materials. But they can be stored either in the local system or in some other places on the Web.

**Inference engine**: this is an important component of our system. The engine allows querying the knowledge base; it reasons and infers a new knowledge, new facts from the given knowledge in the knowledge base. When a user sends a request, it takes the query and accordingly to execute, it collects the facts from the knowledge base (respective ontologies and rules) and stores them in the working memory. After processing the query based upon the given facts, it infers the knowledge and sends response to the user through the user interface. In developing this central component of our system, we used, Euler [12], a reasoning tool (discussed in chapter 4).
Service layer: it is the service layer, where different services are offered. The services are, such as, personalized search services, querying services, navigation, learning material annotation services, notification services, etc.

User interface: it is a common integrated user interface managed by the rights controller. As per the users’ privileges, the rights manager gives access to the system. For example, a student can search and navigate the learning materials, can receive notifications, can edit his own profile, but s/he will not be able to change other students profile, cannot have the write permission to the knowledge base, etc. While the tutor will have the write permission to the knowledge base, s/he can update the ontologies, can add new learning materials, can annotate the learning materials, etc. Again, between the tutors and administrators, the privileges can be set as per their defined roles.

5.8 Ontology Framework
In this section we present the ontology frameworks developed to create personalized learning environment. The ontologies developed are modular. The advantages of modular based ontology is that it is easy to manage and easy to incorporate the new set of concepts within the ontology at any given point of time.

The main components of our modular ontology are domain ontology, document ontology and student ontology. The following sections discuss them in detail.

5.8.1 Domain Ontology
Domain ontology models the concepts and relationships of a particular subject area to be taught instead the general concepts. Wikipedia [13] defines domain ontology as, 'ontology that models a specific domain, or part of the world. It represents the particular meanings of terms as they apply to that domain'. Particularly in an eLearning scenario, domain ontology
describes the course contents semantically as it is built in poly-hierarchical structure with more formalized relations between the subject concepts.

5.8.1.1 Step Followed in Building Domain Ontology
Here that we considered “Indexing and abstracting” as an instance of a course to be taught under the program “Masters in library and information science”. So here our aim is to build the domain ontology for the course “Indexing and abstracting” for the purpose of demonstration of the system. In order to build the domain ontology on a subject area “indexing and abstracting”, we followed the following steps.

Step 1: Search for an existing ontology
In building domain ontology, it is always worthwhile that we should search for the existing ontology on that domain. It reduces our burden in terms of identifying and selecting a set of terms requires in building the ontology and also sensitizes about the relationships of those terms. Unfortunately we did not find any existing ontology on “indexing and abstracting”.

Step 2: Extraction and accumulation of the terms
The next step is to extract the terms. In this regard, we first consulted with the existing Knowledge Organization Systems (KOS), such as, Colon Classification (CC), Dewey Decimal Classification (DDC), Universal Decimal Classification (UDC) and Library of Congress Subject Heading Lists (LCSH). Unfortunately, we did not get much help in terms of a source of subject terms from these traditional KOS tools even though they helped in boosting our work. We believe it is because we aimed in going deep into micro-area whereas these traditional KOS aim to cover the entire world universal knowledge at more generic levels. So they cover broader part of the subject areas instead in depth. We consulted with the text books, research articles, reports, glossaries, etc, in extracting the terms.
Step 3: Structure
In this step we decide about the structure of the ontology. There are several issues involved in designing the structure of the domain ontology. The design issues are discussed in the section 5.8.1.2.

Step 4: Standardization of subject terms and their relationships
In this step we standardize the subject terms using tools like thesauri or some other controlled vocabulary tools. Another important thing in ontology is to identify and formalize the object properties (concept relationships) and the date type properties (attributes).

Step 5: Ontology development
Based on the output of steps 1-4 above, we finally build the ontology.

The following figure 5.4 represents an excerpt of the terms used in domain ontology on “indexing and abstracting”. It shows the basic structure of our domain ontology which is designed following the facet approach [14]. It is worth to mention that this basic structure of the domain is finally transformed into a standard ontological model (discussed in section 5.8.1.3).

As we see from the following figure 5.4, the child elements of root concept “information and retrieval” is grouped into two based upon the characteristics, namely, “byProcess” and “byProduct”. Candidate terms under the category “byProcess” are “indexing” and “abstracting”. Again “indexing” is further subdivided into two “defined” and “assigned” based upon the characteristic “byType” and so on. On the other hand, the candidate terms under the category “byProduct” are “indices” and “abstracts”. The candidate concepts under the concept “indices” are further categorized into two, “byMaterial” and “byType”. The figure also shows that, there can be related terms of a candidate term. The related terms are to establish the associative relations between the concepts, neither the hierarchical nor the equivalent relations between the concepts. For example, as shown in the figure, “indexing” is related to “thesauri” and “filling
systems”. It is also possible that a concept can have instances which is expressed in the following figure by “kind-of” relation. For example, the instances for the concept “pre-coordinate indexing” would be, “Chain indexing”, “POPSI”, “PRECIS”, etc.

In formalizing the domain ontology, we transformed these characteristics (by process, by product, isKindOf, etc.) into the standard properties. The details of the properties (relational and attributes) are discussed in Appendix A.

Figure 5.4: Excerpt of the indexing and abstracting domain terminologies
5.8.1.2 Design Issues

The most important issue in building the domain ontology is its design framework. Will it be enough if we just organize the domain concepts in a taxonomic (genus/species) structure or do we need to do something more? It is observed that in many cases the domain ontology are modeled simply by establishing the "is-a" relationship which provides the genus/species relationships among the concepts [15]. The problem with such type of simple "is-a" concept relationship is, it stops in achieving many of the ontological advancements. Ontology is more than just taxonomy. "is-a" relationship does not express exclusively the relations between the concepts and often leads to misrepresentation [16].

Another important issue we had to consider was, in order to define the classes and the semantic relations between those classes, do we create them from scratch or do we follow some standard which is already in the place and then extend (if needed) to meet our system requirements? The work carried out in [15, 16] built the classes and semantic relations by their own from scratch. But we preferred to use a standard framework already in place. The advantage in using standard is it will be community understandable, easy to share with others and reusable. Also it can be readily accessible to the third party software agents already configured using a standard. This way we can easily overcome the issue of interoperability.

5.8.1.3 Formalization of Domain Ontology

In order to formalize the domain ontology we mainly used the Simple Knowledge Organization System (SKOS) framework [17]. The rationale behind using SKOS is, it is an area of work that develops specifications and standards to support the extraction and use of knowledge organization systems (KOS) such as thesauri, classification schemes, subject heading systems and taxonomies within the framework of the Semantic Web. It provides a standard way to represent knowledge organization systems using the Resource Description Framework (RDF). Encoding the information in
RDF allows it to be passed between computer applications in an interoperable way which to be used in distributed, decentralized metadata applications. Now a day’s decentralized metadata is very common to service providers. Because they want to add value to metadata harvested from multiple sources [18].

Figure 5.5 shows a snapshot of the domain ontology framework consisting partial classes (in elliptic shape) and properties. There are two types of properties defined, namely, object property and datatype property. object properties (e.g., skos:hasTopConcept, skos:broader, etc.) represents the semantic relation between two classes and datatype properties, i.e. attributes (e.g., skos:note, skos-ext:hasInstance, skos-ext:synonym etc.) whose values are plain literal. The classes and properties with prefix skos represent the classes and properties and are taken from SKOS whereas the classes and properties with prefix skos-ext are the extensions of SKOS defined locally to fulfill the system requirements. The detailed description
of the classes and properties of domain ontology are provided in Appendix A.

It is worth to point that we considered a minimal and sufficient set of classes, semantic relations and attributes in building the domain ontology. In considering the classes, semantic relations and attributes, we emphasized the requirements and goals of the system. Considering the eLearning system as a self-learning space, we captured all the related information that would be required in satisfying the learner requirements. We also tried to capture the relevant information that would be required by the tutors, course creators, etc.

5.8.1.4 Namespace

The namespaces used in the domain ontology are shown in table 5.1 and in table 5.2. In addition, the namespace for domain ontology is,

\[ \text{dom: http://localhost/domain#} \]

<table>
<thead>
<tr>
<th>Namespace-prefix</th>
<th>Namespace</th>
<th>Schema used</th>
</tr>
</thead>
<tbody>
<tr>
<td>skos</td>
<td><a href="http://www.w3.org/2008/05/skos#">http://www.w3.org/2008/05/skos#</a></td>
<td>SKOS</td>
</tr>
<tr>
<td>skos-ext</td>
<td><a href="http://localhost/SKOS-EXT#">http://localhost/SKOS-EXT#</a></td>
<td>Locally developed Schema extending SKOS</td>
</tr>
</tbody>
</table>

Table 5.2 shows the common namespaces used in all the other ontologies as well, i.e. document ontology and student ontology.

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Namespace</th>
<th>Schema used</th>
</tr>
</thead>
<tbody>
<tr>
<td>rdf</td>
<td><a href="http://www.w3.org/1999/02/22-rdf-">http://www.w3.org/1999/02/22-rdf-</a></td>
<td>RDF</td>
</tr>
</tbody>
</table>
5.8.2 Document Ontology

Document ontology is to model and to describe the learning contents and their component parts semantically. It is envisioned to simulate the learning object internal structure (discussed in detail in section 5.6) as we see in real world into an ontological form for semantic computation. It comprises a set of classes and sub-classes, semantic relations (object properties) and attributes (datatype properties).

5.8.2.1 Formalization of Document Ontology

Figure 5.6 shows the main building blocks of our document ontology. In the diagram the *italic classes* represent the “abstract” classes whereas the non-italic classes are the “concrete” classes. Idea is, “abstract” classes cannot have any instances directly whereas the “concrete” classes can have instances directly.

Class *Entity* has four subclasses: *Program, Course, LearningObject* and *Contribute*. *Program* refers to a knowledge area which represents customized courses. A set of courses are tailored to a particular program. A *Course* is a sequence of various topics. Class *LearningObject* describes learning resources, in our case digital materials including web-based materials that can be used and re-used to support learning. For example, multimedia content, instructional content, course objectives, instructional software and software tools, persons and organizations, or events referenced during technology supported learning. Whereas, class *Contribute* (lom-life:Contribute) defines the entities (i.e., people, organizations) contributing to the state of the learning objects during its life cycle (e.g., creation, edits, publication).
The class LearningObject consists of three subclasses, such as, Topic, ContentUnit and SupportResource. Topic represents the theoretical explanation about a topic of knowledge that is complete in the scope of the intended explanation. A set of topics are available to a course. The relation between Topic and Course is established by the property “isAvailableToCourse” and its inverse property is “courseHasPart”. One of the most important characteristics of our approach to document ontology is, it is modeled in such a way where the bibliographic information and item specific information can be expressed separately, whereas most of the existing models on document ontology do not provide such provision. Here, class Topic comprises a set of properties which describes the bibliographic information. For example, dc:title, dc:creator, dc:contributor, dc:subject, lom-gen:keyword, hasPrerequisite, isPrerequisiteOf, etc. While class ContentUnit comprises a set of properties for describing the item specific information, such as, lom-tech:location (of the learning objects), requireNCSpeed (require network connection speed), lom-tech:format, etc. The relation between the classes Topic and ContentUnit is established by the property “hasContentUnit” and its inverse property “isContentUnitOf”.

Class Topic is divided into two, such as, Atom and Module. An Atom is the simplest learning material which deals with a single "learning concept" for example, an HTML page that deals with a single learning concept "Java Inheritance". A Module is relatively complex learning object consisting of several chapters or sections or subsections or all. A Book, Book chapters, Journal articles, a single Web page (or HTML file) that might incorporate one or more sections, images, etc are examples of a Module. A Module is further divided into three subclasses namely Chapter, Section and Subsection. Chapter defines the chapter of a learning object (e.g., book chapter), Section defines the sections within a chapter whereas Subsection represents the subsections within a section of a learning material. The classes Chapter, Section and Subsection are related by the properties hasPart and isPartOf.
Class *ContentUnit* is used to describe the content fragments. The properties of *ContentUnit* are, *hasLearningObjective*, *isVersionOf*, *hasVersion*, *isContainedOf*, *lom-tech:format*, *lom-tech:location* and *requireNCSpeed*. It is worth noting that, *hasVersion* holds the information of a content unit available in different formats (e.g., a textual content may have other forms, such as, audio, graphic, etc.). In order to uniquely identify the different forms of content units, we divide the *ContentUnit* class into five subclasses (Figure 5.6), such as, *Text* (e.g., textual content of the materials, such as, letters, dissertations, poems, newspapers, articles, archives of mailing lists),
*Dataset* (e.g., lists, tables, databases), *Image* (e.g., images and photographs of physical objects, paintings, prints, drawings, other images and graphics, animations and moving pictures, film, diagrams, maps, musical notation), *Software* (e.g., Java/C source files, MS-Windows, executables, or Perl script) and *Sound* (e.g., an audio compact disc, and recorded speech or sounds). Further granularity of these classes, *Text*, *Dataset*, *Image*, *Software*, and *Sound* are shown in figure 5.7. The class *Dataset* is further divided into three, *Databases, Lists* and *Tables*. The *Image* class is divided into *MovingImage* and *StillImage*. *MovingImage* is further divided into three subclasses, *Animations, TelevisionPrograms* and *Videos*. *StillImage* is divided into five subclasses, *Drawings, GraphicDesigns, Maps, Paintings* and *Plans*. The class *Sound* is divided into two, such as, *Music* and *Speech*.

![Content unit ontology class diagram](image)

**Figure 5.7: Content unit ontology class diagram**

The class *SupportResource* as shown in figure 5.8 represents the resources used as a support (reference) document by an individual learner. These resources are to be made available to a student only when the course
instructors recommend them. The class SupportResource is divided into three subclasses, Document, Tool and Links. Class Document points to learning materials, such as, Articles, Books and Slide presentations, etc. Class Links points to the recommended site URLs and class Tools identifies the recommended tools needed by the students during their learning activities. The class Document is divided into three, Article, Book and SlidePresentation. The class Article is further divided into two, namely, JournalArticle and NewspaperArticle.

Figure 5.8: Support document ontology class diagram
The other classes and subclasses of document ontology are shown in figure 5.10. The class Contribute is divided into two, Entity (lom-life:Entity) and Role (lom-life:Role). Class Entity is used to identify any information about entities (i.e. people, organizations) contributing to the learning objects whereas the class Role specifies contributors such as, author, publisher, illustrator, editor, graphical designer, instructional designer, etc. The classes lom-edu:Context, LearningObjectives and lom-edu:LearningResourceType brings the context into the learning space. The class lom-edu:Context identifies to whom the resource is intended for/ useful. For example, the learning materials could be useful for higher education, such as by graduate and post-graduate students.

Class LearningObjective represents the educational objectives of the resources, for instance, define, evaluate, introduce, example, compare, classify, demonstrate, describe, design, etc. We have identified and
accommodated a total of 19 concepts qualifying educational objectives of the learning objects. In identifying the educational objectives we consulted the Bloom’s taxonomy of educational objectives [20]. This is a unique approach in terms of modeling and using the same for contextual information retrieval in a distributed environment.

Class lom-edu:LearningResourceType identifies the potential educational use(s) of content associated with the learning resource. It is divided into two subclasses, such as, EducationalResource (e.g., example, exercise, index, etc.) and ExaminationResource (e.g., exam, project_task, questionnaire and self-assessment).

The above three classes, namely, lom-edu:Context, LearningObjective and lom-edu:LearningResourceType are representing the context layer of the conceptual framework for semantic web driven eLearning system (discussed in section 5.6 ).

Figure 5.10: Visual representation of document ontology classes using OWL Viz
The class *dc:Language* identifies the language of the learning materials (e.g., eng, hin, ben, etc.) whereas as the class *LanguageScheme* points to the language scheme (e.g., ISO 639-2). Class *lom-edu:Difficulty* identifies difficulty/competency level required to work with the resource. We defined five difficulty/competency levels, namely, easy, very_easy, medium, difficult and very_difficult. The class *lom-edu:IntendedEndUserRole* is used to define the principle user(s) for whom the learning object is designed, for example, Author, Learner, Manager and Teacher. Class *lom-tech:Format* assigns the technical datatype(s) of (all the components of) the learning object. It is narrowed down into two subclasses, namely, Continuous (e.g., audio, video) and Discrete (e.g., text, image, application).

The detailed description of the classes, semantic relations (object properties) and class attributes (datatype properties), constraints, etc. are provided in Appendix B.

### 5.8.2.2 Namespace

The namespaces used in the document ontology are shown in table 5.3. In addition the common namespaces which are used in document ontology are shown in table 5.2. The default namespace of document ontology is,

$$: \text{http://localhost/doc-onto}#$$

<table>
<thead>
<tr>
<th>Namespace-prefix</th>
<th>Namespace</th>
<th>Schema used</th>
</tr>
</thead>
<tbody>
<tr>
<td>dc</td>
<td><a href="http://purl.org/dc/elements/1.1/#">http://purl.org/dc/elements/1.1/#</a></td>
<td>Dublin core</td>
</tr>
<tr>
<td>dcterms</td>
<td><a href="http://purl.org/dc/terms/#">http://purl.org/dc/terms/#</a></td>
<td>Dublin core metadata terms</td>
</tr>
</tbody>
</table>

Table 5.3: Namespaces for Document Ontology
Document ontology is developed following the ontological model as discussed in section 5.8.2.1. It is already stated that all the ontologies are built in OWL-DL and in developing and validating them we used Protégé ontology editor. We chose Protégé because of its simplicity, easy user interface, open source and the most popular ontology editor in the domain of open source ontology editors. It has a big base of users’ community. Due to its large user community base, there are many plug-ins available. Also the large user community spread throughout the world provides continuous support.

Following figures provide some of the instances of our document ontology in Protégé pane.

Figure 5.11, displays the document ontology classes along with the description of each class. As we can see from figure 5.11, class Course, a subclass of class Entity, has a description on the right hand side (RHS) using a RDF Schema property rdfs:comment. From the following figure we can also see that there is a constraint description for class Course which is basically a necessary condition. The constraint specifies that each course can be described by providing maximum 4 keywords. So, here the maximum cardinality is set to 4.
Figure 5.11: Document ontology classes in protégé pane

Figure 5.12 displays list classes of document ontology in the extreme left hand side (LHS). The figure also displays the in-build text editor in the foreground of the figure which allows in modifying the value of the properties, such as, rdfs:comment, rdfs:label, etc. of a class.
Figure 5.12: Classes of Document ontology in Protégé pane

Figure 5.13 shows the properties, both object properties (a) and datatype properties (b) of document ontology in Protégé pane.
The following figures show the class instantiation and their description pane. Figure 5.14 shows the instantiation window for sub-class Text of class ContentUnit of the learning materials. Figure 5.15 shows the instantiation window for sub-class EducationalResource of class lom-edu:LearningResourceType. Similarly, figure 5.16 shows the instantiation window with the instances for class LearningObjectives.
Chapter 5: eLearning Services Using Semantic Web Technologies

Figure 5.14: Class instantiation in Protégé pane
Figure 5.15: Class instantiation in Protégé pane
5.8.3 Student Ontology

Student ontology is designed to describe the students’ community and representing the related concepts. It captures the students’ information. In general we can call it as ontology based learners’ profile. It is an important component for any ontology driven eLearning system as one of our mottos is to develop the learner centric (*personalized and community driven*) eLearning system. It is being considered as a background knowledge provider about a student to the system. In fact, student ontology is extremely important in service point of view while we look for satisfying the learners learning goals more meaningfully (contextually).

The described information can be categorized into two, such as, generic information about the learners and the other one is about their educational information. Generic information includes student name, their gender, contact address, language, in terms of proficiency and disability in a language, speed of network connectivity and so on. Whereas academic
information includes the course registered for, their learning goals, their style of learning, cognitive style of learning, competency, subject interests, qualification, their education level, affiliation and so on.

### 5.8.3.1 Formalization of Student Ontology

In formalizing the student ontology we present here a framework of student ontology, shown in Figure 5.17.

![Class diagram of student ontology](image)

Figure 5.17: Class diagram of student ontology

The framework is developed centering the class `Student`. The other classes and sub-classes are built surrounding the `Student` class. Along with the class
Student, its properties (object and datatype) are shown in figure 5.17 with their range. To make it further clearer a small instance is provided in figure 5.18. In figure 5.18, dotted ellipses and triangles are to specify the range for the properties (both for object and datatype) of class Student.

![Figure 5.18: Partial class diagram of student ontology](image)

The class Course is shown in the above figure 5.17, actually not directly a part of student ontology; rather that is a part of document ontology. To specify which course the student is registered for, we use the class Course. This class primarily sensitizes the system about the learners learning goals while generating the services. The property registeredForCourse creates a semantic relation between the classes Student and Course. To define student background knowledge we assign an attribute hasKnowledgeOn. The domain and range of this property are class Student and xsd:string.
Understanding the learners and their characteristics is an important aspect of any mode of the teaching learning process, be it a face-to-face classroom teaching session, distance mode of learning or eLearning space. Every student has their own way of learning. They have their own way of thinking, processing, building and storing knowledge. The class \texttt{CognitiveLearningStyle} defines cognitive (i.e., ability to link one thing with another) characteristics that the student possesses. Student cognitive styles can be inferred by conducting the tests, like Ross and Witkin tests [21]. The cognitive characteristics formalize the type of information processing and reasoning the student uses. These properties are useful in user modeling, so that contents can be tailored to fit each individual student’s characteristics. We defined five instances of student cognitive learning style, such as, \texttt{Analogue-Analytic}, \texttt{Concrete-Generic}, \texttt{Deductive-Evaluative}, \texttt{Relational-Synthetic} and \texttt{Indefinite style}.

In a similar way, each student has their own style of learning. For example, there are students who prefer to learn by reading materials while some students prefer to learn through graphics or some by just listening. There are also types of students who learn better by doing hands-on practices, play with the software and tools. They prefer to learn by doing things and so on. The class \texttt{StudentType} captures the variations in student learning styles. To identify the varieties we use Fleming’s VARK model, the most popular and relatively simple model that covers all the aspects of a learner’s learning styles [22] (discussed in detail in chapter 2). The model characterizes the learners into four categories that expand upon Neuro-linguistic programming (VAK) models:

1. \textit{visual learners} - have a preference for seeing (think in pictures; visual aids such as slides, diagrams, handouts, etc.)
2. \textit{auditory learners} - best learn through listening (lectures, discussions, tapes, etc.).
3. \textit{reading/writing learners} – prefer to learn by reading/ writing the textual materials
4. *kinesthetic learners or tactile learners* - prefer to learn via experience—moving, touching, and doing (active exploration of the world; science projects; experiments, etc.).

The class such as, `ims-lip:QCL` identifies the Qualifications, Certifications and Licenses (QCL) of a student. The property `degreeGrantedBy` (identifies the organization that offered the degree) associates the `ims-lip:QCL` class to the `Organization` class. The class `EducationLevel` identifies the student level of learning. `Interest` class identifies the student interests.

The `Language` class enumerates the student languages into three different levels, expressed by the properties, `languageProficiency`, `languageDisability` and `languageEfficiency`. Here, by proficiency we mean that the student has great competency on language(s). Efficiency means that the student has skillful knowledge or balanced knowledge or moderate knowledge on a particular language(s), but he does not have great competency in that language. Whereas, disability defines that the student has disability towards a particular language(s). By dividing student known language into three different levels, we tried to personalize the system as well as we tried not to confine the system capacity under a very strict rule. Here, we tried that even if materials are not available in a language in which student is competent, the system still try to retrieve those materials in a language in which student has balanced or moderate knowledge instead of just not delivering the materials or deliver a material in which student does not have language knowledge. For example, student A is proficient in Marathi, efficient in English and has disability in Bengali. Now in the document corpus there are documents only in English and in Bengali. In this case student A will receive the documents in English not in Bengali. The system by keeping the language information in three different levels, is actually avoiding the risk of either no recall or high recall language-wise.

The `Competency` class identifies the student competency or skill level. The class `Time` which is applicable to the entire model identifies the temporal
information and is divided into four subclasses, namely, Day, Month, Season and Year.

The detailed description of the classes, semantic relations (object properties) and class attributes (datatype properties), constraints, etc. are provided in Appendix C.

### 5.8.3.2 Namespace

The namespaces used in student ontology are shown in table 5.4. In addition common namespaces which are used in student ontology are shown in table 5.2. The namespace for student ontology is,

```
stu-onto : http://localhost/student#
```

<table>
<thead>
<tr>
<th>Namespace-prefix</th>
<th>Namespace</th>
<th>Schema used</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcard</td>
<td><a href="http://www.w3.org/2001/vcard-rdf/3.0#">http://www.w3.org/2001/vcard-rdf/3.0#</a></td>
<td>vCard</td>
</tr>
<tr>
<td>ims-lip</td>
<td><a href="http://localhost/~bisu/ims-lip/1.0#">http://localhost/~bisu/ims-lip/1.0#</a></td>
<td>IMS-LIP</td>
</tr>
</tbody>
</table>

Student ontology is developed following the ontological framework as discussed in section 5.8.3.1. Like other ontologies, student ontology is also built in OWL-DL and in developing and validating it we used Protégé ontology editor.

Following figures provide some of the instances of student ontology developed using Protégé.

Figure 5.19 shows the student ontology in Protégé pane. The axioms we see in the following figure are the axioms defined for the Student class in the knowledge base. For example,

```
stu-onto:educationLevel exactly 1
```
It restricts the value of educationLevel of the student into 1 (cardinality is 1). That means a student at a time cannot be both an undergraduate and a graduate student. S/he can be a student of only one education level at a time. Similarly, the following axiom says that the value of a property typeOfStudent for a Student should be at least one (some) from class StudentType.

\[
\text{stu-onto:typeOfStudent some stu-onto:StudentType}
\]
Figure 5.20 shows the properties in Protégé pane for the Student ontology. The left most (a) are the object properties, whereas the right most (b) are the datatype properties.

Figure 5.21 presents the object properties for student ontology in Protégé pane. It also shows the domain and range of a domain. For example, the following figure shows that for a property stu-onto:hasCognitiveLearningStyle, the domain is stu-onto:Student and range is stu-onto:CognitiveLearningStyle class.
Similarly, figure 5.22 shows the datatype properties in Protégé pane used in student ontology. Whereas, figure 5.23 shows the Student class instantiation and description window in Protégé.
5.9 Learning Design

In this section we discuss learning designs and expressing them in logic rules. These are based upon the system competency as discussed in section 5.3. These logic rules are encoded using the N3Logic rules [23] (discussed in detail in chapter 4). N3Logic uses N3 (Notation 3) syntax [24] and extends RDF with a vocabulary of predicates. N3Logic is a subset of First Order Logic (FOL). As N3Logic is a subset of the FOL, it is more expressive and is useful as a tool in the open Web environment. [23]. Rules are built to reason over distributed information sources (ontologies). Here we present some sample rules as developed for our project.
Case 1: Learning style (student type)

We discussed in section 5.8.3.1 that there are students who prefer to learn by reading materials while some students prefer to learn through graphics or some by just listening and so on. We also discussed that to identify the varieties, we use Fleming’s VARK model, the most popular and relatively simple model that covers all the aspects of a learner’s learning styles [22].

In order to provide the documents based upon the student type (learning style), we created a class, called StudentType in the “student ontology”. Four instances are created for the class StudentType, such as, visual; auditory; reading-writing and kinesthetic. In the document ontology, we have a class, called, lom-tech:Format which is further divided into two, namely, Continuous and Discrete. For the class, Continuous we defined two instances, such as, audio and video and for the Discrete class we have defined instances, such as, text, application and image. The format is assigned for each content unit for the topics. lom-tech:format property is assigned for the stated purpose where the domain and range of it is ContentUnit and lom-tech:Format respectively.

To implement it, we match the student type with the document format. For example, if a user has reading-writing habit, then it is understood that s/he learns through the textual materials. Similarly, the auditory learners prefer to learn through the audio materials. So, we match and retrieve the document tagged with format “audio” to the auditory students. In the same way, visual learners with the “video”, and “image” materials while the kinesthetic learners with the “application” related materials.

```prolog
@forAll S, D, C, V.
{S a stu-onto:Student; stu-onto:typeOfStudent stu-onto:reading-writing. C lom-tech:format text} =>
{S :eligible_to_receive C}.
```
Here, S is a student and has reading-writing habit. According to the above discussion, here the condition is, if a student has reading-writing habit, s/he must be given the textual documents only. We see here that the learning items (C) with textual format should be provided to the reading-writing learners. In order to meet the individual learning goal the learner searches through subject term. During the search time the system takes care of the individual student learning style as it recognizes the learner once the learner logs into the system. The query could be formulated like the following. In the query we call the conclusion of the above rule as premises as shown below.

\[
@\text{forAll } S, C, D, V.
\{ D \text{ dc:subject dom:Pre-coordinate_indexing;
hasContentUnit } C. C \text{ hasLearningObjective define. } S
\text{ :eligible_to_receive } C \} \Rightarrow \{ S :\text{receive_with } C \}.\]

It is to be noted here that, in order to meet the context based search, the learner can set the learning context from the interface during the search time. As we see here, the learning objective is “define”. Similarly the learners can assign the other learning context (as discussed in section 5.8.2.1) during the search time.

The following rules show the rules for other three types of learners, such as, auditory learner, visual learners and kinesthetic learners respectively. For visual learners we created two rules since we identified two types of document format, “video” and “image” for visual representation of the learning contents.

\[
\{ S \text{ a stu-onto:Student; stu-onto:typeOfStudent stu-onto:auditory. } C \text{ lom-tech:format audio} \} \Rightarrow \\
\{ S :\text{eligible_to_receive } C \}.\]

\[
\{ S \text{ a stu-onto:Student; stu-onto:typeOfStudent stu-onto:visual. } C \text{ lom-tech:format video} \} \Rightarrow \\
\]
The following figure shows the screen shot of the executing result of the query as mentioned above against the rules for all four types of learners. We used “Euler” reasoning engine in executing the query. From the following figure 5.24, we see that the system infers the search result for student_1, and student_3, two reading-writing students receive all the textual materials, student_2, an auditory student receives the audio learning materials, and student_4, a visual student receives the visual materials.
Case 2: Cognitive learning style (CLS)

CLS defines cognitive characteristics that the student possesses. These can be inferred from cognitive tests as the Ross and Witkin tests [21]. The cognitive characteristics formalize the type of information processing and reasoning the student uses. These properties are useful in user modeling, so that contents can be tailored for each student’s characteristics. We identified five instances as student cognitive learning style, such as, Analogue-Analytic, Concrete-Generic, Deductive-Evaluative, Relational-Synthetic and Indefinite style.

In order to model CLS, in the student ontology we have a class called, student:CognitiveLearningStyle and defined five instances of it as discussed above. Now consider a case, for example, a student with “deductive-evaluative” style of learning has learning goal “pre-coordinate indexing”. This type of student prefers to learn first “pre-coordinate indexing” by taking the theory first and then practice. Whereas the concrete-generic type of students with the same learning goal prefer to learn by taking the examples first and then figure out the theory.

In order to meet the above goal we created a class called lom-edu:LearningResourceType which is further divided into two, such as, EducationalResource and ExaminationResource. Under the EducationalResource the instances are, example, exercise, lecture, index, problem_statement, simulation and tutorial; whereas, for ExaminationResource the instances are, namely, exam, project_task, questionnaire and self-assessment.

In the following rule example, we show the rule made for delivering the learning materials to fulfill the learning goal on “pre-coordinate indexing” suiting the individual learner CLS. Here we consider a student with “deductive-evaluative” learning style. The condition is, s/he must receive the lecture materials first followed by exercise materials.
@forall S, D, scp, C, L1, L2, L3, sub.

{S a stu-onto:Student; stu-onto:hasCognitiveLearningStyle stu-onto:Deductive-Evaluative. (scp 2) e:findall (C {D dc:subject dom:Pre-coordinate_indexing. D lom-edu:learningResourceType lecture. D hasContentUnit C} L1). (scp 2) e:findall (C {D dc:subject dom:Pre-coordinate_indexing. D lom-edu:learningResourceType exercise. D hasContentUnit C} L2). (L1 L2) list:append L3} =>

{S :WORKING_deductive_student_get_list L3}.

Here, S is a student with deductive-evaluative learning style. D is a topic with content unit C (for details see section 5.8.2.1). L1 is a list of lecture materials whereas L2 is a list of exercise materials. L3 is a list containing the lists L1 and L2 maintaining the order L1 followed by L2.

It is worth to mention here that, we use built-in property, “findall” (e:findall) from Euler [11], an inference engine supporting logic based proofs. The syntax of e:findall is as follows,

((?SCOPE ?SPAN) e:findall (?SELECT ?WHERE ?ANSWER))

It unifies ?ANSWER with a list that contains all the instantiations of ?SELECT satisfying the ?WHERE clause in the ?SCOPE ?SPAN of all asserted n3 formulae and their log:conclusion.

The following figure shows the execution result for the deductive-evaluative students against the query. In the knowledge base, only Student_1 and Student_2 are marked as the deductive-evaluative student.
Similarly, for other type of cognitive learning style learners, such as, Analogue-Analytic, Concrete-Generic, Relational-Synthetic and Indefinite style the learning design is made.

Case 3: Background knowledge
It is important that when the student takes a new topic for study, the system must check whether we have the required knowledge to take this new topic. In case of face-to-face class room teaching, a teacher knows his students background, so he can deliver accordingly or before starting a new topic, he can provide the background of the new topic. In case eLearning system, it is an important issue. The system must be designed in such a way that while a student searches for a topic, the system should be able to check whether the
searched topic has any prerequisite and if yes, then the system must check student background knowledge to see whether the student fulfills the prerequisite knowledge. In order to do this, we defined a datatype property (attribute) called, \textit{stu-onto:hasKnowledgeOn} whose datatype is \texttt{xsd:string} in our student ontology. Similarly in mentioning the learning material prerequisite(s), we create a datatype property called \textit{hasPrerequisite} whose datatype is \texttt{xsd:string} in the document ontology.

In designing this situation in the system, we assigned the following rule.

\begin{verbatim}
@forAll S, T, D, C, P.
{S a stu-onto:Student; stu-onto:hasKnowledgeOn T. D hasPrerequisite P; hasContentUnit C. P string:containsIgnoringCase T} => {S :Pre_receive C}.
\end{verbatim}

The rule says, a Student S, has knowledge on topics T. Document D has prerequisite(s) P and the documents has content unit C. Now, if the document prerequisite(s) P is in the student background knowledge list T, then the student is eligible in receiving the learning materials. It is worth to mention that the \texttt{string:containsIgnoringCase} is a build-in function of N3Logic.

An example query could be like the following.

\begin{verbatim}
@forAll S, D, C.
{S :Pre_receive C. D dc:subject dom:Pre-coordinate_indexing. C hasLearningObjective define} => {S :Q_8_receives C}.
\end{verbatim}

The query says that, 'retrieve learning materials those are on \textit{definition} of “pre-coordinate indexing” '. Against this query, the following figure 5.26 shows the search results.
In the knowledge base, it is defined that to retrieve the learning materials on definition of pre-coordinate indexing, the learners must have the knowledge on indexing system. In our knowledge base only Student_1 and Student_3 has knowledge on indexing system. So, we can see from the above figure that the materials retrieved for Student_1 and Student_3 only.

**Case 4: Education level**

Every document has its target audience. So, when a student searches for a learning resource, the resource must be same level of the student education level. The idea is when a student is a primary level student; we cannot just simply deliver the resources made for advance level students. To design and implement the same into our system, we created a subclass called,
HigherEducation, under a class lom-edu:Context in our document ontology. Two instances are defined for the class HigherEducation and they are Undergraduate and Graduate. Learning resources are annotated according to their education level.

Similarly in our student ontology, we have a class called stu-onto:HigherEducation under a class stu-onto:EducationLevel. And then two instances, stu-onto:Undergraduate and stu-onto:Graduate are created under the class stu-onto:HigherEducation. Students are annotated as per their education level.

So, here the condition is an undergraduate student always receives the resources made for undergraduate students. In a similar way, a graduate student always receives the resources made for the graduate students. The following example shows the design for the undergraduate students.

\[
@forAll S, D, C. \\
\{S a stu-onto:Student; stu-onto:educationLevel stu-onto:Undergraduate. D dcterms:educationLevel Undergraduate; hasContentUnit C\} \Rightarrow \{S :\text{under-graduate_student_eligible_to_receive C}\}.
\]

In the above rule, S is a type of undergraduate student. D is a document whose target audience is undergraduate students. Document D has content unit C. For above rule, the system infers the result as shown in the following figure 5.27 against the following example query where the search term is mentioned as “Pre-coordinate indexing”.

\[
@forAll S, C, D. \\
\{D dc:subject dom:Pre_coordianate-indexing; hasContentUnit C. S :\text{under-graduate_student_eligible_to_receive C.}\} \Rightarrow \{S :\text{Q_4-2_under-graduate_student_receives C}\}.
\]
In the knowledge base, Student_2 and Student_3 are the undergraduate students. The figure shows Student_2 and Student_3 receive the materials whose target audiences are undergraduate students only. To show the same on the screen, we mention the target audiences at the end of the file name of the learning materials as we see in the above figure 5.27.

In a similarly way, we design the same for the graduate students as shown below.
Case 5: Learners language and learning resource language

Another important aspect is language of the student as already mentioned. When the system delivers a learning resource to a student, the system must verify the student language proficiency and the language of the resources available.

In implementing this we created a class called `dc:Language` and instantiated it following the standard ISO 639-2 [25]. The instances are, for example, “eng”, “hin”, “mar”, “ben”, etc. Then we annotated the documents as per their languages using the property `dc:language` of document ontology whose `domain` is class `Entity` and `range` is class `dc:Language` respectively. Similarly, we annotated the students as per their respective languages using the property `dc:languageProficiency` of student ontology where the `domain` and `range` of this property are, `stu-onto:Student` and `dc:Language` respectively.

We present here a simple snapshot of our design in solving the issue. Here, S is a type of student, and has language proficiency in English. The condition is, English proficient student receives only the materials (C) those are in English.

```plaintext
@forAll S, D, C.

{S a stu-onto:Student; stu-onto:languageProficiency eng. D dcterms:educationLevel Graduate; hasContentUnit C} => {S :graduate_students_eligible_to_receive C}. 
```
Chapter 5: eLearning Services Using Semantic Web Technologies

The result for the above scenario is shown in the following figure 5.28 against the following sample query.

@forAll S, D, C.

\{S :receives_doc_mar C. D dc:subject dom:Pre-coordinate_indexing; hasContentUnit C} => \{S :Q_6-2_marathi_lang_proficient_students_receive C\}.

Figure 5.28: Search result for the students who are proficient in Marathi language

The system retrieves the materials in Marathi language for Student_3 as he is the only student in our knowledge base whose language proficiency is Marathi.
It is worth to mention that the example shown above considered a simple scenario whereas, in real sense it is much more complicated. For example, a student is proficient in English language. So, the condition is only those resources will be provided to this student which are in English language. Now suppose, in repository there are no resources available in English language, in that case the recall fails.

To handle this situation, we created another two types of instances, where we define in the student profile, about the student “language sufficiency” and “language disability”. By “language sufficiency”, we mean that the student has sufficient knowledge on that language(s) to use a resource whereas, by “language disability”, we mean the student has no knowledge on a particular language(s). So, the system works against a query of a student as per the following algorithm,

1. Take a query and collect the student language proficiency
2. Check whether the available learning materials language match with the student language proficiency list.
3. If yes, then retrieve the materials, and go to step 10.
4. List the language(s) of the available materials in the working memory.
5. Match that list of language(s) against the student language sufficiency list
6. If any match found, then process and retrieve the materials and go to step 10.
7. Match the list with the student language disability list.
8. Discard all the materials available in matched language(s) list
9. Retrieve the materials in the languages which are available in the student disability language list  
   ## Note: It may possible that the student is neither efficient nor disable in a particular language(s) and may have a little knowledge on them. Since we don’t have the option in capturing such kind of instances, at this point we considered to retrieve those materials not matched with the student disability language list.
10. Terminate the process

Case 6: Network connection speed of student and bandwidth requirement of the learning materials

Network connection speed of a student is another important aspect in context of eLearning. Though now a days’ Internet connection speed is not a big issue, but still it has an important impact when we deal with the multimedia materials. For example, in the present system we considered visual type of students, which means these types of students prefer to learn through the visual representation of the learning contents, such as, videos and images which really need high bandwidth for transmission over the network.

So, here the condition is, the network connection speed of a student must match with that required by learning material(s). In order to design this, we created a property, requireNCSpeed, whose domain is ContentUnit and range is xsd:string and enumerated a list of instances (128 Kbps, 256 Kbps 512 Kbps, 1Mbps, etc.) in the document ontology. Similarly, in the student ontology created a property called, stu-onto:hasNCSpeed whose domain is stu-onto:Student and range is xsd:string and enumerated a list of instances (128 Kbps, 256 Kbps, 512 Kbps, 1 Mbps, etc.). The implemented rule is shown below.

@forAll S, D, C, P, P1.
{S a stu-onto:Student; stu-onto:hasNCSpeed P. D hasContentUnit C. C requireNCSpeed P1. P list:in P1} => {S :compliant_NCSpeed_with C}.

The rules says that a student (S) has network connection speed is P. Similarly, a material C requires network connection speed is P1. Now, if the student network connection speed P belongs to the learning material
requires network connection speed P1, then, student S receives the materials.

Case 7: Network connection speed: provision of alternative version

We extended this further considering that if the student network speed does not match with that required by the resource, in such case, the system looks for alternative versions which may match with the student network connection speed. It is true that this is against to the idea of learners learning style (student type) (Case 1), but still we prefer to provide this as we understand that when there is no match, instead of recall failing, we can provide some alternative versions which actually deals with the same learning content but maybe in different formats. For example, a visual learner has network connection speed 128 Kbps, and a material (e.g. a high resolution image) s/he looks for, requires 256 Kbps speeds. In this case the system looks for the alternative version for the same learning material (say, textual material describes the same learning content) which may need the network connection speed equal to the learners network connection speed. So, the system over rides the rules of learners learning style (student type) and delivers the materials. In is worth to mention that, we assume that we are dealing here with the physically normal students, not any other exceptional students, for example, physically challenged students, etc. In case of those exceptional cases, we may need to stick with our rules as discussed in Case 1. But it needs to be explored further. However, in our present study we did not deal with such cases, we may take this aspect in our future work.

The following rule deals with the situation where the system looks for an alternative format of a learning material. It is worth to mention here that, this rule activates only, when the previous rule fails.

In order to implement this, we created a property called, \textit{hasVersion} whose \textit{domain} and \textit{range}, both are \textit{ContentUnit}. It is to be noted here that, except the “item” related information, such as, \textit{resource format, resource location},
size, require network connection speed, etc., the “bibliographic” information remains the same as these are the alternative versions of the actual learning contents for a topic.

@forall S, D, C, V, P, P2.
{S a stu-onto:Student; stu-onto:hasNCSpeed P. D hasContentUnit C. C hasVersion V. V requireNCSpeed P2. P list:in P2} => {S :compliant_NCSpeed_with_version V}.

Here, S is a student and has network connection speed P. Learning content C has alternative versions V and the required network connection speed to access those in a list, P2. Now, the condition is, if a student network speed matches with the list of required speed of the alternative versions of resources that originally require higher speed, the system retrieves the alternative.

In a similar way, we addressed the other system perspectives as discussed in section 5.3 from two different perspectives, such as, student perspectives and tutors/administrators perspectives.

5.10 Conclusion
In this chapter we presented the semantic eLearning system developed. We presented here the framework and semantic learning layer cake which provides the foundation in building and generating semantically rich Web based information services. We argued in favour of fine-grained descriptions of learning contents and modular based knowledge representation techniques. We presented our layered system architecture with the related technologies and tools used in each layer of the system. It is shown that fine-grained semantic description of learning contents using semantically rich languages can improve the present eLearning state. We demonstrated 7 cases which show the effectiveness and efficiency of the proposed system. We would like to further state that the area of semantic
operations in digital repositories is akin to the demonstrated system discussed here. This may be explored in future extensions of the present work.

5.11 References


18. Introduction to SKOS. http://www.w3.org/2004/02/skos/intro