Ontology serves as an information centre and acts as a heart for Semantic Web technology. It also serves as a domain knowledge base with concept instantiations. Domain information is incorporated in the Ontology through classes, objects, individuals and data type values, both object and data properties. Ontology is appropriate for designing the model of the knowledge domain. Ontology facilitates machine readable data semantics. It also facilitates searching, interchange and unification of domain knowledge. Ontology building is regularly done with definite task in mind which helps in restricting the content and framework of the Ontology. There are various tools for Ontology development which can be applied to Ontology life cycle. The stages in Ontology life chain are the development, population, implementation and sustainment of Ontology [23].

Common understanding of a domain and related information is shared by Ontology. This sharing can be among humans or among software agents. Ontology development process thoroughly analyzes the domain knowledge. So, an excellent knowledge of the domain is provided by Ontology. Also, it promotes reuse of knowledge thus saving time and effort [24].

Under Knowledge Management System, knowledge classification is regarded as Ontology. In Ontology, the facility of searching is provided with the functionality of semantic searching. So, it is different from keyword based searching as provided by traditional search engines. Also, searching functionality of Ontology is more efficient than traditional methods of searching. Important concepts and their properties description for each concept in a domain are described in a hierarchical fashion in the Ontology [5]. Ontology, thus, contains the following:

- Classes: Representation of concepts either physical or abstract (i.e. specific or conceptual) in a taxonomic structure (i.e. hierarchical structure). Classes define hierarchy in super class- sub class manner.
- Relationships: Representing the associations (usually binary) between concepts.
• Attributes (Properties): Describing features of concepts.
• To model the sentences which are always true, formal axioms are defined.
• Instances (Individuals): To represent the elements in Ontology.

3.1 Ontology Development Tools

The first stage towards Ontology development was to search an appropriate tool for its development. There are benefits from tool support for Ontology development, as this process is largely domain dependent and complex also. In spite of the availability of enormous Ontology development tools freely on the web, they rarely achieve their objectives. Protégé, OilED, Apollo, RDFedt, Swoop are among various tools available for Ontology development which were chosen for comparative analysis. In order to identify the most suitable tool, comparison of above mentioned Ontology development tools becomes imperative based on certain features. Comparison features chosen were modeling features, limitations, base language, merging, information extraction and lexical support as shown in Table 3.1 mentioned under Section 3.1.5 [25]. The main purpose of these tools is Ontology development but they also provide feature like merging and integration of Ontology, annotation tools, storing and querying etc. During the analysis stage of Ontology development tools, several different aspects were found. There is a move towards java platforms and extensible architectures. Ontology tools are lacking in interoperability as well as storage in databases [6]. Protégé tool for Ontology development was found to be the domain independent and dominant tool. It provides graphical user interface and provides storage through JDBC and many features which are not provided by other tools. Its popularity is due to the availability of enough documentation on web and is extensible also.

3.1.1 Protégé

Protégé tool for Ontology building acts as a knowledge base framework and an open source Ontology editor. It supports platform for Ontology modeling via web client or desktop client. Through Protégé, various formats like OWL, RDFS and XML schema are available for building Ontology. It can be considered as a flexible base for application development as well as rapid prototyping because of its provision of plug-
and-play environment, extensible facility and base language as Java. Developers community, academicians, government users, corporate users are using this tool in the diverse areas such as biomedicine, education, intelligence gathering, corporate modeling. So, this tool is supported by strong community [26].

### 3.1.2 OilED

OilED is an Ontology editor for the construction of OIL based Ontology. OIL is Ontology in Infrastructure for Semantic Web. It helps in creation as well as editing of OIL Ontology. It is developed to demonstrate the use of DAML+OIL. DAML language is the extended version of XML and RDF. With DAML+OIL language, Ontology development takes Object Oriented approach. OilED does not support Ontology development environment fully, large scale ontology building, migration and integration of Ontologies, versioning and augmentation which are involved in building Ontology. Also, no extensibility is provided. Though, class expressions, concrete type expressions, essential and defined classes can be used [6]. FaCT system provides reasoning support which is incomplete for OIL which is extended with data types and individuals. It does not include additional services such as explanation [27].

### 3.1.3 Apollo

In Apollo, Basic primitives like classes, instances, functions, relations and many other are used for modeling Ontology. OKBC protocol is the basis for the internal model which actually is a frame model. In this, Ontology is organized hierarchically. Inheritance of Ontology from other Ontology is allowed and they can act as parent Ontology also. Every default Ontology contains primitive classes. Instances created by each class inherit all slots (consisting of set of facets) of the class [6]. This tool does not support collaborative working but can be extended with extra plug-ins. It is Java based and provides features like consistency check, Ontology storage in the form of files, import/export format [28]. But, it does not support web information extraction, graph view, collaborative processing and multi-user capabilities.

### 3.1.4 RDFedt

Building of complex and structured RDF as well as RSS is allowed in RDFedt. Overview of complicated data structures is provided with fundamental trees. Facility of additional functions is available which helps in testing data, giving comments and provision of error messages. This tool supports RDF, RDFS and Dublin core
elements. This tool for Ontology building serves as an editor for textual language but is not based on Java. It is platform dependent and only works with windows operating system [6].

### 3.1.5 Comparative Analysis of Ontology Development Tools

Protégé is the utmost leading and domain independent tool which was found in an online survey [15] and was used by 75% respondents of the survey. Availability of online help is the major reason for such enormous developers tends towards Protégé for Ontology development.

<table>
<thead>
<tr>
<th>Features</th>
<th>Protege</th>
<th>OilEd</th>
<th>Apollo</th>
<th>RDFedt</th>
</tr>
</thead>
<tbody>
<tr>
<td>Import format</td>
<td>XML, RDF(S), XML schema</td>
<td>RDF(S), OIL, DAML+OIL</td>
<td>OCML, CLOS</td>
<td>RDF(S), OIL, DAML, SHOE</td>
</tr>
<tr>
<td>Export format</td>
<td>XML, RDF(S), XML schema, FLogic, CLIPS, Java, HTML</td>
<td>RDF(S), OIL, DAML+OIL, SHIQ, dotty, HTML</td>
<td>OCML, CLOS</td>
<td>RDF(S), OIL, DAML, SHOE</td>
</tr>
<tr>
<td>GUI</td>
<td>Via plugins like GraphViz and Jambalaya</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Consistency check</td>
<td>Via plug-ins like PAL and FaCT</td>
<td>Via FaCT</td>
<td>Yes</td>
<td>Only checks writing mistakes</td>
</tr>
<tr>
<td>Multiuser</td>
<td>Limited (multiuser Capability added to it in 2.0 version)</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Web support</td>
<td>Via Protégé-OWL plug-in</td>
<td>Very limited name spaces</td>
<td>No</td>
<td>Via RSS</td>
</tr>
<tr>
<td>Merging</td>
<td>Via Anchor-PROMPT plug-in</td>
<td>No</td>
<td>No</td>
<td>N/A</td>
</tr>
<tr>
<td>Collaborative working</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Ontology library</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Inference Engine</td>
<td>With PAL</td>
<td>With FaCT</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Ontology storage</td>
<td>File &amp; DBMS (JDBC)</td>
<td>File</td>
<td>Files</td>
<td>Files</td>
</tr>
<tr>
<td>Extensibility</td>
<td>Via plug-ins</td>
<td>No</td>
<td>Via plug-ins</td>
<td>No</td>
</tr>
<tr>
<td>Availability</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
<td>Free</td>
</tr>
</tbody>
</table>
So, Protégé tool was used for the development of IRSCSD. Comparison of various Ontology development tools is shown above in Table 3.1.

### 3.2 MDA Standards for Ontology Development

Ontology development was performed by applying MDA standards. In the initial phases of Ontology development, UMLs were used for depicting domain specific knowledge as it is based on object-oriented paradigm [29]. MDA standards were followed for Ontology development under Object Modeling Group.

Architecture of MDA comprises of four layers where each layer has its own separate set of defined standards. The four layers are shown in Figure 3.1 below. Top most layer M3 and is called as Meta-Meta Model layer. MDA’s Meta Model layer is the M2 layer. Third layer is M1 layer and is called as Model layer. The bottom layer is M0 which is an Instance layer.

<table>
<thead>
<tr>
<th>Layer Nomenclature</th>
<th>Layer Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M3 Layer</td>
<td>Meta-Meta Model</td>
</tr>
<tr>
<td>M2 Layer</td>
<td>Meta Model</td>
</tr>
<tr>
<td>M1 Layer</td>
<td>Layer Model</td>
</tr>
<tr>
<td>M0 Layer</td>
<td>Layer Reality</td>
</tr>
</tbody>
</table>

![Figure 3.1: Layers of Model Driven Architecture](image)

Various standards are defined for each separate layer. M3 Layer is Meta-Meta Model layer at the top of the MDA architecture consisting of group of concepts which are used for defining Meta-Models. OMG’s standards are defined at top most layer and are called as MOF. For modeling a language, model used is called as Meta Model [29]. Ontology development language defined at this layer is OWL/RDF language i.e. Ontology web development/resource description framework. At the M2 Layer new Meta-Model is defined which covers some specific application domain. Like in Ontology development for IRSCSD system, Computer Science has been taken as one of the application domain. UMLs used at this layer represent Meta-Models. Real-world models are developed at M1 Layer on the basis of domain specified. In this layer all the classes, relations, states etc. are developed. Instances of the concepts
defined at the model layer M1 are at the bottom most layer i.e. M0 Layer. Figure 3.2 below shows the modeling architecture based on MDA for Ontology development.

So, M0 Layer is the real world which includes all possible things which are represented using models at M1 Layer. Representation at M1 Layer is abstract and simplified, depending on how rich these models are. Concepts defined in Meta-Models are used for defining models. So, each Meta-Model determines expressivity of its models. M2 is the layer which represents the location of the meta-models [29]. The meta-models are also defined using some concepts. The separate M3 Layer at the top of MDA architecture called as Meta-Meta model layer acts as a residence for the set of concepts which are used to define Meta-Models at M2 Layer.

This MDA architecture comprises of models and Meta-Models based on:

- Object-oriented Meta-Meta Model like MOF
- Ontology
- Semantic Web technologies
3.3 Three-layer Design Principles for Ontology Development

Three layer design principles were used in Ontology development and were named as Top Layer, Middle Layer and Bottom Layer as shown in Figure 3.3. Quality of Ontology is influenced by the design principles at middle and bottom layers [24]. Design principles were majorly for layers at middle and bottom level and top layer design principles were based on overall Ontology development.

3.3.1 Design Principles at Top Layer

Ontology is a kind of software program. This top layer focused on the overall Ontology building process complaint with traditional software development process. Software development process is based on SDLC techniques. Similarly, for Ontology development, process was followed systematically which included various phases like determining the domain and scope of Ontology, source identification, class creation, identification of properties, creation of individuals and consistency check (Details mentioned in Section 3.4.3).

3.3.2 Design Principles at Middle Layer

Middle Layer handled the explanation of generic guidelines and constraints through which major steps as well as their ordering was specified. Under Middle Layer, there were three major principles identified for designing Ontology:

Design Principle 1

As Ontology development process is independent of problems related to terminology, so focus was given on concepts rather than terms. While, developing Ontology for IRSCSD system, focus remained on elemental conceptual Structure of the target domain. Mixed approaches for the development of Ontology were followed which included top-down, bottom-up and middle-out rather than sticking to the single approach.

Design Principle 2
In the starting phase of Ontology development, identification of top-level category was done for governing rest of the steps. Main components and attributes of concepts like “part-of” relation were identified. After completing is-a hierarchy formation and informal term definition, axiom writing was done.

*Design Principle 3*

Only after building is-a hierarchy, detailed information of concepts and terms was provided. Issues related to the arrangement of terminology were resolved at the last step. Each term corresponds to exactly one concept in Ontology as a well-organized conceptual structure is build rather than a dictionary. Is-a hierarchy building is on high priority than term definition.

### 3.3.3 Design Principles at Bottom Layer

Bottom Layer consisted of minute level guidelines such as those for class identification. Three major design principles were followed at this layer:

*Design Principle 1*

Determination of essential properties of concepts and instances was done. The essential properties of super classes were inherited through is-a relationships. One attribute of super class was used in distinguishing sub class and super class.

*Design Principle 2*

Multiple inheritance relations were avoided as much as possible. Basic concepts and role concepts were clearly differentiated. Ambiguity regarding boundaries between similar concepts is not considered much.

*Design Principle 3*

When terms were found with multiple meanings then new terms are created. New terms were added when not able to find appropriate term for a concept. Representation issue is dealt carefully. Consultation of an upper Ontology was done, when distinction of categories for both general and high level were required.

### 3.4 Integration of Ontology Development Techniques for IRSCSD Framework

Various Ontology development techniques which included: tools, standards and design principles were finalized for domain Ontology construction. Second phase of
IRSCSD system was Ontology building for Computer Science domain. For this, Protégé tool was finalized as described in section 3.1.5. Also, MDA standards and top-down approach were followed for construction of Ontology. Focus was laid on the layers of MDA at each step of Ontology creation. Figure 3.4 shows how various techniques for Ontology development (tools, standards and guidelines) were integrated with Ontology development phase of IRSCSD system.

3.5 Discussion

With tools, techniques, standards, design principles identified and incorporated, IRSCSD system was ready for the design and development of Ontology. Therefore, design and development of the IRSCSD system’s Ontology have been explained in detail in Chapter-4.