Chapter 5

Domain Modeling

Increasingly, people seem to misinterpret complexity as sophistication, which is baffling.... the incomprehensible should cause suspicion rather than admiration. Possibly this trend results from a mistaken belief that using a somewhat mysterious device confers an aura of power on the user.

Niklaus Wirth.

This chapter describes the technique of knowledge representation for a subject domain. After a brief introduction, it reviews the related work in the area of domain modeling in the context of intelligent tutoring systems. Described next are the features of XML followed by the proposed model of XML based subject domain. The chapter concludes with the summary and discussion.

5.1 Introduction

Knowledge representation developed as a branch of artificial intelligence in general and expert systems in particular. One of the components of expert system called 'knowledge base' needs an effective method of knowledge representation for the domain of expertise so that the 'inference engine' can use it for reasoning and problem-solving phase [Rolston, 1988]. Knowledge representation has a long tradition rooted mainly in philosophy, logic, and psychology. It is the application of logic and ontology to the task of constructing computable models for some domain.

According to Davis [Davis et al., 1993], there are five basic principles about knowledge representation:

A knowledge representation is a surrogate.
Physical objects, events, and relationships, which cannot be stored directly in a computer, are represented by symbols that serve as surrogates for the external things. The symbols and the links between them form a model of the external system.

*A knowledge representation is a set of ontological commitments.*

Ontology is the study of existence. For a database or knowledge base, ontology determines the categories of things that exist or may exist in an application domain. Those categories represent the ontological commitments of the designer or knowledge engineer.

*A knowledge representation is a fragmentary theory of intelligent reasoning.*

To support reasoning about the things in a domain, a knowledge representation must also describe their behavior and interactions. The description constitutes a theory of the application domain.

*A knowledge representation is a medium for efficient computation.*

The knowledge is encoded in a form that can be processed efficiently on the available computing equipment.

*A knowledge representation is a medium of human expression.*

A good knowledge representation should facilitate communication between the knowledge engineers who understand AI and the domain experts who understand the application.

In the context of an intelligent tutoring system, knowledge representation for a particular subject has traditionally been named as 'domain modeling'. Although this nomenclature in the first instance may appear little different, it is appropriate in the realm of intelligent tutoring as the purpose of knowledge representation in expert systems and knowledge representation in intelligent
tutoring systems is different. In an expert system, an inference engine scans the knowledge base for reasoning and provides solution for a given problem [Rolston, 1988]. In an intelligent tutoring system, the course generation module and the tutor module interact with the domain module for selecting the next topic to be sequenced to the learner. Thus, no inference mechanism is involved. While various techniques of knowledge representation (like production rules, semantic networks, frames and scripts) for domain modeling in an intelligent tutoring systems have been and can be borrowed from AI literature, the requirements of knowledge representation in ITSs are different than in an AI system. This is described in the next section.

5.2 Framework of Domain Model

Domain modeling can be based on the premise that learning resources in conventional educational set up are organized as a hierarchy. Learning resources are divided into many subjects according to the institution. One subject is divided into many units; one unit is divided into many atomic materials and examinations. Once this hierarchy is defined, sequence is automatically determined for the subjects, units, topics down to the materials and tests. This hierarchy is followed in top-down manner during design phase, while it is used in bottom up fashion during execution phase. However, learning activity will not complete only with the transfer of these materials. Skills of absorbing, recalling, joining, applying and selecting learnt information are the other important issues in learning activity. These skills are called 'learning strategy'. Learning strategy is domain dependent, but some strategy is reusable to other domains. Moreover, navigational skill of teachers is the part of information of learning activities. Skill of selecting, joining, communicating and emphasizing the learning material is termed as "instructional
strategy". This strategy is thought to be relevant to the learning strategy, and also be domain dependent. This instructional strategy can be implemented as some specific framework of navigation. SCORM specification [SCORM, 2004] has a straightforward and simple navigation flow, but it will be more flexible to reflect the navigation framework based on both learning and instructional strategies. When these strategies are identified as a part of a domain model, domain modeling will have a new way to accelerate its productivity and quality. Even if learning materials of different subjects are developed, the navigation and feedback framework can be reused if these subjects share the common learning or instructional strategy. This is the basic theme used in this research for both tutor modeling and domain modeling.

Specifically, the representation of a subject domain knowledge is motivated from the lesson planning activity of a human tutor. A human tutor plans her teaching activity from the curriculum, the learning objectives, the time duration, the learners background and the available course material. Although teaching strategy may differ from subject to subject, a common thread can be identified in planning activity. To truly mimic the behavior of a human tutor, her lesson plans serve as good guidelines for domain knowledge representation. Using this framework of lesson planning by a human teacher, following requirements for domain modeling are identified:

i. Subject independence,

ii. Simplicity,

iii. Ease of dynamic lesson generation, and

iv. Ease of lesson sequencing and presentation.
5.3 Related Work

To explore knowledge representation techniques, the standards for learning technology were reviewed for applicability. The IEEE Learning Technology Standards Committee (LTSC) [IEEE, 2004] study groups cover many topics. One group is developing the Learning Object Model (LOM). This standard proposes metadata to describe a learning object. The metadata allows it to be located, as a card catalog helps to locate a book at a library. Due to its intent to describe all learning objects, it does not have recommendations for content. The metadata elements pertain to publishing concerns and is very little of educational concern for learners and teachers at all levels.

Existing markup languages were considered next. Many good characteristics and drawbacks as well, were determined during the survey of markup languages [OASIS Cover Pages, 2000]. It was discovered that there are very few markup languages used to document the development of instructional content. Although there are many markup languages for different domains, only seven (7) of the 450 referred to learning, teaching or education!

The Instructional Management Systems (IMS) metadata project [IMS, 2004] and ADL Sharable Content Object Reference Model (SCORM) [SCORM, 2004] deal primarily with the metadata to describe the learning objects. Only two refer to the educational content directly. These are the Learning Material Markup Language (LMML) by Süß [Süß et al., 2000] and the University of Bristol’s Tutorial Markup Language (TML) [TML V.4, 2002]. Two additional languages were located as well. These are SGML/XML used by the Open Learning Agency and Open School Project (OLA/OSP) [Klassen et al., 2002], and the RLO/RIO concept from Cisco Systems [Cisco Systems, 2000]. These four were reviewed for
use. Tutorial Markup Language (TML) has content written in XML, processed by Perl scripts to generate the webpages for NetQuest, at the Institute for Learning and Research Technology at the University of Bristol [NetQuest, 2004]. The software has been developed both for authoring and for displaying TML documents. It is designed to provide the structure for learning experiences taking the form of expository information, followed by practice questions, a drill or an evaluation, and a conclusion. This is appropriate for fact-based knowledge. SGML/XML, as described by Klassen et al. at OLA/OSP, presents a structured framework to describe core curriculum courses [Klassen et al., 2002].

The LMMML language was developed by Christian Süß [Süß et al., 2000]. It is based on modular XML, with data definitions for the major parts of on-line learning and instructional environments generated as HTML. It appears that the logic is incorporated in the content. Modular XML includes only those files that it needs to perform the transformations. LMML contains document type definitions (DTDs) with the elements designed for specific domains of learning materials.

The Reusable Learning Object strategy (RLO) from Cisco Systems [Cisco Systems, 2000] is used to train people to use particular products and perform particular processes. It is a strategy of using smaller pieces called Reusable Informational Objects (RIOs) that vary in content based on the information they will contain (such as concept, fact, process, principle, or procedure information). Each RIO is composed of content, practice and assessment material. Each type of RIO has guidelines for the minimum information and activity requirements needed to cover the information.

In summary, some of the markup languages reviewed support development of learning material with the granularity of a lesson, while others mimic textbooks
on-line in producing products in HTML. A summary of the findings of the review of markup languages is outlined in Table 5.1. It is clear that a new markup language needs to be developed that is capable of constructing learning objects (LO) and providing the structure for reusable objects. It needs to be designed to use education based terminology, to build a structure for the storage of information for instructional content, instructional events, system functionality, and sequencing of the learning experience from lesson planning to tutoring environment. The proposed Domain Markup Language (DML) is designed to meet these requirements.

Table 5.1. Comparison of various Markup Languages and Standards

<table>
<thead>
<tr>
<th>Language</th>
<th>Features</th>
<th>Drawbacks</th>
<th>Desirable Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOM/SCORM</td>
<td>Metadata.</td>
<td>Not used for educational content description.</td>
<td>Include instruction and teaching-specific metadata.</td>
</tr>
<tr>
<td>HTML</td>
<td>Static web pages; easy to use in the Classroom, independent of platform.</td>
<td>Not easily reused; Incorporates navigation and lesson control.</td>
<td>Output product in HTML, but store content, style and logic in different structures.</td>
</tr>
<tr>
<td>JSP, ASP, PHP</td>
<td>Dynamic functionality with &quot;types of pages&quot; and variable content.</td>
<td>Typically a one-pass process with logic and style together.</td>
<td>Develop the logic and style to be applied independently. Produce &quot;types of objects&quot; consistent with the LO attributes and functionality.</td>
</tr>
<tr>
<td>RIO/RLO LMML TML</td>
<td>Standardized Content.</td>
<td>Includes element labels from extraneous domains in DTDs; Limited to teaching facts, skills and processes; Size too large for easy reuse.</td>
<td>Use lesson planning and teaching terminology, consistent with the LO design. Develop structure for experimentation, exploration and experiences.</td>
</tr>
</tbody>
</table>
Table 5.1. Comparison of various Markup Languages and Standards (Continued.)

| SGML | Course or Unit templates used for consistent design; Built-in alternative materials. | Many of the terms based mostly on presentation style information. Incorporates logic with style. | Use templates of the most common set of teaching tasks and teaching objectives. Develop LOs to use agent, web services and servlet technologies. Incorporate the alternative materials structure. |

Why not use other schemes of XML such as LOM [IEEE, 2004]? We do utilize certain concepts from LOM indirectly in our design. LOM uses an XML schema to describe the information about existing learning material and does not describe the content. The content may be programmed in any number of different formats, and contains learning materials described as courses, lessons, or blocks of lessons. It does not facilitate the use of the material except in its entirety.

If the components are coded separately we have greater flexibility. By using the components, then the material may be more easily reused. An example of this is "instead of linking the entire encyclopedia, let's provide just the topics for the lesson."

We generate the most useful educational content by using subject-independent markup called Domain Markup Language (DML). The language is based on the educational activities that are performed in the process of planning, organizing, assembling and delivering the content. This is not intended to be exclusive, or cover all fields, but it would produce a readily sharable and reusable set of educational resources useful to many fields of study.
5.4 Review of XML

This review is mainly adapted from [Bradley, 2000; Goldfarb & Prescod, 2003; Zeid & Gupta 2001]. XML or eXtensible Markup Language is a text based markup language that allows storing data in meaningful and structured way. It is extensible, allowing users to define their own tags with their own attributes and values. These values provide semantic qualifications to data and context. XML provides context validation at both the semantic and syntactical levels. It checks for completeness and well-formedness of documents. XML also supports rich structures, similar to those found in object oriented programming, and database applications. XML is a metalanguage. XML is simply a set of rules for creating new document types.

XML defines XML documents. XML documents define data objects. XML is based on the well-known object oriented paradigm. XML documents provide access to the content and structure of the data objects. An XML document uses tags to define both the data and structure of the objects that the document defines, and eventually creates. The structure of the data defines the object hierarchy. XML tags can be nested. They may have attributes. Attributes may have values. XML tags follow strict rules. They are case sensitive, white space matters, and all attribute values must be enclosed in quotes.

The logical structure of an XML document consists of declarations, tags (elements), comments, characters, and processing instructions. All of this logical structure is indicated in the document by explicit markup. The logical structure of an XML document refers to its tags, their structures relative to each other, and their semantics. Tags come in two groups. One group has tags that start and end. The other group has empty tags. The logical structure must be well formed and valid.
The well-formedness and validity requirements are used by XML parsers (processors) during processing XML documents. It is much easier to check the well-formedness of a document than checking its validity. All what we need to do to check if a document is well formed, is to check its tags against XML syntax rules. Checking the document validity requires us to establish the semantics rules that we use as a reference to measure the document semantics against. These semantics are included in the DTD (document type definition). Thus, we need to write a DTD for every XML application we create. XML parsers validate XML documents using the DTDs. A DTD contains statements that define to the parser what is possible to do in a valid XML document that uses this DTD. Each statement looks like a tag. XML parsers read all DTD statements before they begin parsing a document.

XML documents could be referring to other documents that they may need. The physical structure of an XML document consists of all entities that are contained in the document. The main document represents the “root” or the “document” entity. The document entity serves as the starting point for an XML parser. Entities could be parsed or not. Parsed entities are replaced by their corresponding text that becomes part of the main XML document. Unparsed entities usually refer to a resource whose contents are not text. For example, an executable that is referenced in an XML document is an unparsed entity.

The ultimate goal of writing XML documents is to display and use them on the Web. XML tools must check both the syntax and semantics of an XML document before displaying it. We can identify these tools as parsers and viewers. XML parsers check both the syntax and the semantics. XML viewers display the XML document. Parsers and viewers can come in separate software, or may be...
bundled together. There are two types of parsers: validating and non-validating. If the parser only checks the document syntax, then it is non-validating. This type of a parser only checks the well formedness of an XML document. Non-validating parsers do not use the DTD of the XML document they are parsing. Validating parsers, on the other hand, checks both the well formedness and validity of the XML document. They check both the syntax and semantics of XML documents. These parsers use DTDs while performing the validity checks.

The complete XML model requires two documents: the XML document and the DTD. We pass the two files to a validating XML parser that parses the XML file using the DTD file. If there are syntax or semantics errors, we correct the documents. When the parsing process is successful, the parser may create a data object model (DOM). This model has the object tree of the XML document. We use a viewer to display the XML document. DOM provides a general way of accessing data structures and objects from structured documents. Thus, an application can use XML to manipulate data structures and objects.

5.5 XML Based Domain Modeling

[An]y group that is developing a metadata set is free to limit its work to its narrow interests; it need not take a broader view unless it voluntarily chooses to do so [Milstead & Feldman, 1999]. In line with this remark, we decided to have a separate markup language and based our language design on the concept of Learning Object (LO) attributes. This idea is further strengthened from the specifications suggested in SCORM [SCORM, 2004], IEEE LOM [IEEE, 2004] and Topic Maps [Dichev et al., 2004] which describe the learning contents or web content as composed of sharable and reusable objects that can be aggregated to form more useful objects of interest. The principle of separation of concern is also
kept in mind while designing the new markup language. It states that the conceptual structure of a domain should be separated from the physical structure.

The idea behind domain knowledge representation in the proposed ITS is to have—to the most possible extent—a subject independent domain representation. The subject independence nature will help in designing the domain independent tutor module. Further, the representation should facilitate ‘adaptivity’ of lesson content according to the learners’ knowledge level. The sequencing and presentation of these lessons to the learner are the other issues which can be sorted out easily if the representation is effective.

5.5.1 Efficacy-based Content Classification

One of the most important features of the proposed domain modeling language is the ‘content classification’. While all of the learning material markup languages or the meta-data specifications do have content classification, it is either based on the content type or the learning goal. None of the specifications, to the best of our knowledge, points out the efficacy of the content to enhance a particular cognitive skill of the learner. We advocate the Bloom’s opinion [Bloom et al., 1964] that the memorizing ability, the understandability, the concept comprehension and even the misconception are the cognitive skills that can serve as better indicators of learners’ progress. Bloom’s taxonomy is learner focussed dealing with the types of learning viz. cognitive, affective and psychomotor meaning the knowledge level, the attitude and the skills respectively [Bloom et al., 1964]. However, it provides certain guidelines for instructional design, more specifically guidelines for test design. Based on these guidelines we opine that for achieving ‘adaptivity’ the content should be classified on the basis of their efficacy to strengthen learners’ cognitive level. Consider, for example, one category in
cognitive domain of the taxonomy named as ‘Knowledge: Recall of data’. The examples cited indicate that after a learning experience, the learner can be considered to have acquired knowledge if he can “Recite a policy. Quote prices from memory to a customer. Knows the safety rules”. The keywords at this level are: defines, describes, identifies, knows, labels, lists, matches, names, outlines, recalls, recognizes, reproduces, selects, states.

If these verbs could be used to identify the cognitive level of the learner, they should be useful to classify the learning material. Obviously, this demands much more efforts and expertise on the part of the instruction designer or the content author. The cognitive domain of the taxonomy itself enumerates as many as ninety eight keywords! The content classification at such a micro-level may turn into a futile effort. To remedy this situation, we propose a simple classification based on learners’ observable response. In student modeling, we classified the students’ performance level into four classes. This classification was motivated by the expertise of a human teacher in grading a learner. On the basis of efficacy of the content to strengthen memory, understanding and/or conception twenty three categories are identified. Although the complete design of such a course content is outside the scope of our work, we present in the Table 5.2 the rules to classify the content.

<table>
<thead>
<tr>
<th>Content Class</th>
<th>Memory</th>
<th>Understanding</th>
<th>Conception</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>Low</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>02</td>
<td>Low</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>03</td>
<td>Low</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>04</td>
<td>Low</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>05</td>
<td>Low</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>06</td>
<td>Medium</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>07</td>
<td>High</td>
<td>Low</td>
<td>Low</td>
</tr>
<tr>
<td>08</td>
<td>Low</td>
<td>Medium</td>
<td>Medium</td>
</tr>
</tbody>
</table>
Table 5.2. Rules to classify the content (Continued.)

<table>
<thead>
<tr>
<th></th>
<th>Low</th>
<th>Medium</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Low</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>11</td>
<td>Medium</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>12</td>
<td>Medium</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>13</td>
<td>Medium</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>14</td>
<td>Medium</td>
<td>High</td>
<td>Low</td>
</tr>
<tr>
<td>15</td>
<td>High</td>
<td>Low</td>
<td>Medium</td>
</tr>
<tr>
<td>16</td>
<td>High</td>
<td>Medium</td>
<td>Low</td>
</tr>
<tr>
<td>17</td>
<td>Low</td>
<td>High</td>
<td>High</td>
</tr>
<tr>
<td>18</td>
<td>Medium</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>19</td>
<td>Medium</td>
<td>Medium</td>
<td>High</td>
</tr>
<tr>
<td>20</td>
<td>Medium</td>
<td>High</td>
<td>Medium</td>
</tr>
<tr>
<td>21</td>
<td>High</td>
<td>Low</td>
<td>High</td>
</tr>
<tr>
<td>22</td>
<td>High</td>
<td>Medium</td>
<td>Medium</td>
</tr>
<tr>
<td>23</td>
<td>High</td>
<td>High</td>
<td>Low</td>
</tr>
</tbody>
</table>

The meaning of the rule is like this: A lesson content of class '01' means the content is for the learner whose memorizing ability in previous attempt was tested to be low, the understanding was low and the misconception was high. That means the content has the efficacy to enhance memorizing, understanding as well as the conception.

5.5.2 A Three-layer Domain Model

The proposed architecture for a three-layer domain model is depicted in the Figure 5.1. It consists of a user layer, a conceptual layer and a physical layer. The functionality of each of the layers is described below.

User-Layer: It consists of Lesson Objects (LO) for the presentation of course material to the learner. This is similar to the Sharable Content Object (SCO) as defined in SCORM [SCORM, 2004]. However, its structure is simpler than SCO, and it is obtained after aggregation. It is interfaced with the user through eXtensible Stylesheet Language (XSL).

Conceptual Layer: A meta model that describes a hierarchical structure of the domain. It is the main concern of this chapter. It is similar to the Content
Organization as defined in SCORM. It is a map that represents the intended use of the content through structured units, topics and the lessons. It is this layer that is referred by the tutor module for sequencing the lessons, topics and the units. The relation between a lesson and a lesson object is similar to the relation between a class and object in object-oriented-programming paradigm. While a lesson is an abstract idea, a lesson object (LSO) is a concrete entity composed from the learning assets according to the lesson plan.

Physical Layer: Actual content consisting of text, images, figures, code, facts, principles etc. named as 'Assets' in SCORM. It is the most basic form of a learning resource. Assets are an electronic representation of media, such as text, images, sound, assessment objects or any other piece of data that can be rendered by a Web client and presented to a learner. Assets are stored in XML database and can be updated through XUpdate interface. The retrieval of the assets for aggregation can be done using XQuery language.

The subject domain is composed of several Units. Each unit is composed of several Topics. Each topic being the combination of several Lessons. Thus, a hierarchical structure very similar to a textbook is designed, wherein a unit is equivalent to a chapter, a topic is equivalent to a section and a lesson is similar to a subsection. A lesson is composed of description of the concept with examples and explanation. The physical layer consists of concept description, examples and explanation. The objects at the physical layer are designed in such a way that they can be reused.
Figure 5.1. Three-Layer Domain Model.

The objects at the conceptual layer are the units, topics, and the lessons. The course generator generates lesson objects dynamically for each concept of a topic in a unit. The number of lessons for a topic, the number of topics for a unit and the number of units for a subject domain are fixed by the curriculum designer. As an example, for a typical 64-contact-hour undergraduate course on ‘Operating Systems’, an nine-unit course is suggested as follows:

Unit-I: Introduction; Unit-II: Process Management; Unit-III: Deadlock; Unit-IV: Memory Management; Unit-V: Virtual Memory; Unit-VI: File System; Unit-VII: Disk Scheduling; Unit-VIII: I/O Management; Unit-IX: Case Study.

The advantages of this hierarchical structure are twofold:

i. It allows direct binding of the domain to XML document.

ii. It supports in-built sequencing mechanism for tutor model.

Sequencing first applies to lessons within a topic, to topics within the units and finally to the units. The intended sequencing is defined as part of the conceptual layer, by structuring units, topics and lessons in relation to one another.
and by associating sequencing information with each unit, topic and lesson. The
tutor module is responsible for interpreting the sequencing information described
in the conceptual layer and applying sequencing behaviors to control the actual
sequence of the learning resources at run-time. By following the specifications
from SCORM partially [SCORM, 2004], we intend to support reusability, although
our main concern in designing a separate meta-model for domain representation is
adaptivity and simplicity.

5.5.3 The Domain Model Language (DML)

Given below is an XML DTD based on the proposed DML. It represents a
generic course structure:

<!ELEMENT Subject (Unit +)>

<!ATTLIST Subject
  sname #REQUIRED #PCDATA
  total-units #REQUIRED #NUMBER>

<!ELEMENT Unit (Topic +)>

<!ATTLIST Unit
  tname #REQUIRED #PCDATA
  total-topics #REQUIRED #NUMBER
  unit-num #REQUIRED #NUMBER>

<!ELEMENT Topic (Lesson+)>

<!ATTLIST Topic
  tname #REQUIRED #PCDATA
  total-lessons #REQUIRED #NUMBER
  topic-num #REQUIRED #NUMBER>

<!ELEMENT Lesson EMPTY>

<!ATTLIST Lesson
  resources #PCDATA #REQUIRED
  lesson-num #NUMBER #REQUIRED>
Based on this DTD, a segment of document for the subject "Operating Systems" is given below:

```xml
<Subject name="Operating System" total-units="9">
  <Unit name="Introduction" total-topics="9" unit-num="1">
    <Topic name="Definition" total-lessons="1" topic-num="1">
      <Lesson resources="href="Iso1.doc" " lesson-num="1">
        ...
      </Lesson>
      ...
    </Topic>
    <Topic name="Evolution" total-lessons="6" topic-num="2">
      <Lesson resources="href="Iso1.doc" " lesson-num="1"/>
      <Lesson resources="href="Iso2.doc" " lesson-num="2"/>
      ...
      <Lesson resources="href="Iso6.doc" " lesson-num="6"/>
    </Topic>
  </Unit>
  ...
  ...
</Subject>
```

```
<Unit name="Deadlock" total-topics="8" unit-num="6">
  <Topic name="Definition" total-lessons="1" topic-num="1">
    <Lesson resources="href="Iso1.doc" " lesson-num="1"/>
  </Topic>
  <Topic name="Conditions" total-lessons="2" topic-num="2">
    <Lesson resources="href="Iso1.doc" " lesson-num="1"/>
    <Lesson resources="href="Iso2.doc" " lesson-num="2"/>
  </Topic>
</Unit>
```

Chapter 5: Domain Modeling
... 

</Unit>

...

</Subject>

It is to be noted that the documents for the lessons shown under resources attribute are generated by the courseware generator after getting the content class of the lesson. The courseware generator refers to the ontology-based DTD given below and aggregates the objects belonging to a particular class. The aggregated lesson is stored temporarily as the resource file for subsequent presentation.

The DTD that acts as ontology for the subject as well as the storage for the subject content is defined as follows:

<!ELEMENT OperatingSystem (..., Deadlock, ...) >
<!ELEMENT Deadlock (Definition+, Conditions+, Prevention+, Avoidance+, Detection+, Recovery+, CombinedApproach+, Summary) >
<!ELEMENT Definition #PCDATA >
<!-- COMMENT: Attribute ‘class’ indicates efficacy level, ‘qtype’ indicates question types that can be asked on this topic. ‘rtype’ indicates topic type i.e. concept, example or explanation. ‘otype’ indicates the type of resource -->

<!ATTLIST Definition
class  (01 | 02 | ..) #REQUIRED
qtype  (mcq | trf | fil) #REQUIRED
ctype  (concept | example | explain) #REQUIRED
rtype  (text | image | table) #REQUIRED >
<!ELEMENT Conditions ( MutualExclusion+, HoldAndWait+, NoPreemption+, CircularWait+) >

<!ELEMENT MutualExclusion #PCDATA>

<!ATTLIST MutualExclusion
class (01 | 02 | ..) #REQUIRED
qtype (mcq | trf | fil | na) #REQUIRED
ttype (concept | example | explain) #REQUIRED
rtype (text | image | table) #REQUIRED >

and so on.

The idea behind such representation is that the XML document is used by the adaptive courseware generator that takes input from the student test results, infers the knowledge level of the learner as a 'class', picks up the material of the identified class and aggregates a lesson object. The course generator is described in the chapter on adaptive courseware generation. As shown in the portion of DTD above, the course material has been embedded into a hierarchical structure. Each subject has been divided into units, each unit into topics. At physical layer, the topics are taken as reference to formulate the assets. The lesson objects are actually composed by the course generator from the assets marked up with attributes like concept, example, code, principles, facts and explanation. Each object has been associated with additional attributes: the question type and the resource type. The question type (qtype) may be multiple-choices, true-false or fill-in-blanks. The testware generator can utilize these attributes for formulating the questions.

A portion of the asset embedded into an XML document is shown below:

<Deadlock>
A set of processes is in a deadlock state when every process in the set is waiting for an event that can be caused by only another process in the set.</Definition>

The events with which we are mainly concerned here are resource acquisition and release. The resources may be either physical or logical. However, other types of event like inter-process communication may also result in deadlock.</Definition>

Consider a system with three tape drives. Suppose that there are three processes, each holding one of these tape drives. If each process now requests another tape drive, the three processes will be in a deadlock state. Each is waiting for the event "tape drive is released", which can be caused only by one of the other waiting processes.</Definition>

Consider a system with one printer and one tape drive. Suppose that process P is holding the tape drive and process Q is holding the printer. If P requests the printer and Q requests the tape drive, a deadlock occurs.</Definition>

The underlying philosophy for this hierarchical structure is:

i. Separation of concerns.

ii. Multiple views of the same assets.

The courseware generator is guided by the students' performance level and the tutor model to refer to the conceptual layer, collect the parameters to locate the
required assets from physical layer, aggregate the assets to build a lesson object and present it to the learner at the user layer.

5.6 Domain Model Implementation

The capability of the rapid web development architectures that may be used for education is described in Table 5.3.

<table>
<thead>
<tr>
<th>Table 5.3. Current XML Technology Capabilities for Educational Uses.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Dynamic selection of content (through Xpath, XQuery support) and logic sheets through a web interface (through XSL support) to control the display of content to meet the needs of different audiences,</td>
</tr>
<tr>
<td>2. Storage of information by descriptive metadata making it (through Native XML Databases) searchable and reusable,</td>
</tr>
<tr>
<td>3. Storage of content stored in an XML language based on the domain ontology (through XTM),</td>
</tr>
<tr>
<td>4. Web access to resources, databases and xml-files (through XPath) and</td>
</tr>
<tr>
<td>5. Web Forms and Services (through XUpdate) for easy upload to the server.</td>
</tr>
</tbody>
</table>

This gives us a powerful architecture to produce:

i. A learner-friendly interface where the course content are tailored according to the learners’ knowledge level.

ii. A teacher-friendly interface where the teacher deals mostly with educational and not computer terminology.

iii. A programmer-friendly interface where the development is simplified through several available tools.

Our work is mainly concerned with the first type of interface; there too we only demonstrate the technical feasibility of the proposed model. For identifying a
support tool for domain model feasibility, we carried out an exhaustive search for open source tools and found out a native XML database system named as Xindice [Source-Forge, 2005].

Xindice allows for the rapid development of web environments that use web-server technology and combine the use of a variety of web development languages. Xindice is an open source NXD that uses XML and has a number of characteristics that will enable the development of an e-learning environment from lesson planning content to the delivery of these content in the desired manner to the learner. Logic and display information is necessarily combined within the XSL file for the content to be transformed in one pass. Web architectures such as Xindice allow for the separation of style and logic into multiple XML documents, based on functionality. This is one of the strong points of using NXDs. Educational content can be stored in a form that is both independent of how it appears on a webpage and independent of its functionality. Besides rapid development and reuse, Xindice also allows selective use of documents. The XML-DB Java API provides both update and query interfaces. This allows the generation of semantic-based instructional content from templates and access to the published web page immediately after uploading to the server.

Although the development of a web-site is outside the scope of our work we used Xindice as a basic web server along with the capability of XML document storage. The ontology for subject “Operating Systems” was developed using keywords from the subject domain. We adapted a novel method of ontology design by using the index portion of the books from the domain of Operating Systems. The same simple logic can be used for defining the ontology for any subject. Further, each of the topics can be stored in a separate XML document to achieve fast search and retrieval. Xindice provides facility for index creation based on
elements as well as attributes. This feature can be used for fast retrieval of the document as a whole or portion thereof. The retrieved material is then submitted to XSL filter to prepare it for presentation. The Screenshot 5.1 shows Xindice web server as localhost whereas the Screenshot 5.2 presents its home page in a web browser.

Screenshot 5.1. The Xindice Native XML database Server.

Screenshot 5.2. Xindice Homepage in a client web browser window.
5.7 Summary & Discussion

We run the danger... of creating such complexity that the metadata is effectively useless. Finding the appropriate balance is a central design problem [Lagoze, 1996].

This chapter has first introduced the concept of knowledge representation as applied in expert systems and showed how domain knowledge in the context of ITS differs that from artificial intelligence field. A brief review of knowledge representation techniques has been presented bringing out the shortcomings of the available metadata specifications. A new domain markup language is presented and its basic features are demonstrated. The three-layer architecture supports the principle of separation of concerns and facilitates multiple views of the same learning object.

We present below our explanation regarding the proposed approach.

i. How are the human characteristics incorporated in the model?

The domain model basically represents the curriculum portion of the teacher-curriculum-learner triangle. It is a static entity to be used intelligently by the other humanistic components. Thus, directly there are no apparent human characteristics in the model except for the content organization at the physical layer.

ii. How is the 'adaptivity' achieved?

The adaptivity is achieved through the introduction of 'content class' that reflects the effectiveness of a particular learning object towards the enhancement of the learners cognitive level. As long as the performance remains at a particular level, the same lesson with different content can be presented. The selection of lesson content becomes adaptive to the learners' knowledge level.
iii. Is the 'adaptivity' learner-centered?

As pointed out earlier, the material is organised according to its class which is selected on the basis of the learners' current performance class and hence the model is learner-centric.

iv. How is the 'intelligence' incorporated?

It is incorporated indirectly as a part of the instructional design when a class is assigned to a content.

v. Which of the learning theories advocate the model?

Constructivistic learning theory and cognitivistc learning theory advocate the model. Blooms taxonomy also supports such models.

vi. How is the proposed model domain independent?

The conceptual layer is subject domain independent. The physical layer cannot be subject independent because of ontology. However, no assumption is made about the subject domain nor any subject dependent parameter is incorporated in the model. As long as the subject expert can properly organize the content in a 'class', the given model can be used for any domain.

vii. Is the model technically feasible for web-based implementation?

We have described the feasibility of the model for the subject of Operating system using Xindice. This proves its technical feasibility.

viii. What are the envisaged limitations?

The model demands huge efforts for classification on the part of instruction designer.