

# Design and Implementation of a Multimedia Courseware Database

By

Ali Al-Shammari, B.Sc.

A thesis submitted to  
the Faculty of Graduate Studies and Research  
in partial fulfillment of  
the requirements for the degree of  
Master of Computer Science

Ottawa-Carleton Institute for Computer Science  
School of Computer Science

Faculty of Science  
Carleton University  
Ottawa, Ontario, Canada

31/8/1998  
© Copyright  
1998, Ali Al-Shammari



National Library  
of Canada

Acquisitions and  
Bibliographic Services

395 Wellington Street  
Ottawa ON K1A 0N4  
Canada

Bibliothèque nationale  
du Canada

Acquisitions et  
services bibliographiques

395, rue Wellington  
Ottawa ON K1A 0N4  
Canada

*Your file* *Votre référence*

*Our file* *Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-36878-5

# Abstract

Currently, the area of distance education is gaining high interest, especially in regions where users are geographically scattered and/or influenced by different factors that could affect their learning demands. Distance education puts emphasis on the separation of the learner and the educator in space and/or time.

The emergence of new technologies within different disciplines, such as multimedia, databases and networking, have enriched the learning activities and produced new trends within the distance education era. Telelearning is one of the distance education sub-areas that have obtained increased attention and research.

This thesis discusses a Multimedia Interactive Telelearning System that has been designed and developed at MIRL Laboratory at the University of Ottawa. A new approach has been taken to build a seamless education environment over the Internet. It is mainly based on Object-Oriented paradigm and on utilizing leading-edge technologies. Its aim is to provide real-time interactive multimedia courseware services to a variety of distributed students, and satisfy their learning needs. The system is comprised of several major components: courseware authoring site, courseware database server, multimedia database server, courseware presentation agents, media production center, an on-line facilitator, etc. The accessibility, effectiveness and flexibility are important features of the system that make it a suitable environment for supporting learning services.

The system architecture, courseware data model, database management issues and implementation details are discussed. The main contribution of the thesis is the database modeling, as well as system schema, and the database management architecture.

## **Acknowledgements**

I would like to thank my thesis supervisor - Dr. Ahmed Karmouch for his support, advice and encouragement, in particular, for his valuable comments and suggestions throughout the publication stage.

I am grateful to my parents, family members and my wife for all the support and encouragement given to me throughout the M.C.S. program. It was especially difficult being away from both of my young daughters during these past few months.

I would also like to thank TeleLearning National Center of Excellence for their sponsorship.

# Table of Contents

Chapter 1 Introduction .....	1
1.1 Distance Education: History and Background.....	1
1.1.1 History and Motivation.....	1
1.1.2 Distance Education Models .....	3
1.1.3 Distance Education Participants .....	8
1.2 Terminology Related to Thesis Work.....	11
1.3 Thesis Objective .....	13
1.4 Thesis Outline .....	13
Chapter 2 Multimedia Interactive TeleLearning System (MITS) .....	15
2.1 Objectives and Requirements .....	15
2.2 Generic System Architecture .....	17
2.3 MITS Previous Approach .....	23
2.4 Drawbacks of MHEG-based Model for MITS .....	26
Chapter 3 Description of Courseware Data Model.....	28
3.1 Overview of Multimedia Document Architecture .....	28
3.2 Teaching Architectures .....	31
3.3 Courseware Data Model .....	34
Chapter 4 MITS Database Management Issues.....	38
4.1 Overview of MEDIABASE System .....	39
4.2 Object-Oriented Database Management System .....	40
4.2.1 ObjectStore 5.0 Environment .....	43
4.2.2 PSE-Pro Environment.....	44
4.2.3 Class File Postprocessor tools.....	46
4.3 MITS Database Modeling and Schema Generation.....	47
4.3.1 Relational Modeling .....	49
4.3.2 Object-Oriented Modeling.....	50
4.4 MITS Database Management Architecture.....	57
4.4.1 Client-Server Background .....	57
4.4.2 Description of MITS Database Management Architecture.....	59
4.5 Object Persistency States .....	62
4.5.1 Hollow Persistent Objects.....	63
4.5.2 Active Persistent Objects .....	64
4.5.3 Stale Persistent Objects.....	64
Chapter 5 System Implementation .....	65
5.1 Introduction.....	65
5.2 JAVA Environment .....	66
5.3 Implementation of MITS Database Schema .....	68
5.3.1 MITS Schema Classes .....	68
5.3.2 Schema Evolution .....	72

5.4	Implementation of Application-Server .....	72
5.4.1	Application-Server Modules .....	72
5.4.2	Template for Implementing Additional Modules .....	78
5.5	Annotating MITS Classes .....	79
5.6	Samples of Results .....	81
5.7	System Integration .....	87
Chapter 6 Conclusions .....		90
6.1	Summary .....	90
6.2	Future Work and Suggestions .....	91
7	References .....	94
8	Publications .....	98

# List of Figures

Figure 2.1 MITS's major components.....	17
Figure 2.2 MITS general architecture.....	18
Figure 2.3 MHEG-based model for MITS .....	25
Figure 3.1 Courseware Data Model .....	36
Figure 4.1 ObjectStore5.0 architecture.....	44
Figure 4.2 PSE-Pro architecture .....	45
Figure 4.3 MITS E-R Diagram.....	50
Figure 4.4 MITS OMT Diagram .....	51
Figure 4.5 Primary MITS OMT Diagram .....	52
Figure 4.6 MITS Database Management Architecture.....	60
Figure 4.7 Object Persistency States .....	63
Figure 5.1 MITS Database System Implementation .....	66
Figure 5.2 Application-Server Engine Modules.....	73
Figure 5.3 Template for implementing application-server modules .....	79
Figure 5.4 Original MITS classes and annotated classes .....	80
Figure 5.5 A Snapshot for Creating a Database and Constructing a Courseware .....	81
Figure 5.6 A Snapshot for Constructing a Courseware .....	82
Figure 5.7 Different Generated Databases.....	83
Figure 5.8 The Retrieval of Courseware Meta Data .....	84
Figure 5.9a The Updating of Courseware Meta data.....	85
Figure 5.9b The Updating of Courseware Meta Data.....	85
Figure 5.10 The Retrieval of Courseware After Updating the Database.....	86
Figure 5.11 The Deletion of an Entire Courseware.....	87
Figure 5.12 MITS System Integration Architecture.....	89

# Chapter 1

## 1 Introduction

### 1.1 Distance Education: History and Background

#### 1.1.1 History and Motivation

The terms “Distance Education” and “Distance Learning” have been used interchangeably by various researchers based on their knowledge and backgrounds. “It puts emphasis on the separation of the learner and the educator in space and/or time”, where the volitional control of the learning process is driven by the student rather than the distant teacher [1].

The earliest attempts of distance education date back to the late 1950’s and early 1960’s, when widely broadcasting TV/Radio classes were transmitted to audiences in Europe. This form of educational classes did not provide any sort of interaction between students and teacher, while such interaction is supported in the traditional classrooms. The broadcasting TV/Radio classes also suffered from the lack of bi-directional communications between teacher and students. This reduced the importance of TV and Radio as new resources for educational classes. In the early 1970’s, the emphasis turned from bringing teachers into the classroom to taking students out of the classroom into the outside world [1]. This direction was reversed in the late 1970’s, as newly designed and produced TV series introduced students to new subject matter that was not being taught, yet was considered to be an important complement to the classroom curriculum [1].



Then, in the 1980's, the pendulum swung back to the basics. Meanwhile, the distance learning trend had been affected by communication technologies, such as electronic mail, to provide some sort of interaction between teacher and students. However, researchers did not present a computer-based distance education paradigm for several reasons. For instance, the lack of computer-based distance learning theories and the limitations of multimedia technologies to support complex applications such as distance learning systems.

In the early 1990's, the extensive research in multimedia information processing and enhanced achievements within networking technologies produced a new era of multimedia information applications, which ambitiously aimed to deliver complex multimedia information to distributed users over networks. One of the areas that exploited the high technology growth is distance education. Thus, the education process is shifting from traditional learning style that is based on lecture-and-book model towards non-traditional learning styles where courses are provided through TV classes, CD-ROM and distributed computer systems. The non-traditional learning styles might lead to a totally different way for learners to acquire knowledge in the near future [2].

Currently, distance education is gaining high interest especially in regions where students are geographically scattered and/or involved by a number of factors that could affect their learning demands. Specifically, it is necessary to address the needs of small rural school districts or under-served urban school districts. Also, some university students may need courses to meet graduation requirements that their own universities are unable to offer. For instance, some students enroll in vocational courses; so distance education will be beneficial for them.

## 1.1.2 Distance Education Models

The traditional education model "lecture-and-book" became widely spread since the 1900's. The students are grouped according to their ages and/or experiences, and are gathered together in a classroom so as to learn a specific topic from the teacher's presentation at a specific time. This education model has been efficient for a long period of time [3]. Nevertheless, it is associated with some constraints that could degrade the learning process. One of these constraints is "time", where students have to join a class at a specific time. Another constraint is "space" where learners are supposed to attend a lecture at a specific classroom. When students are geographically scattered and require knowledge transmission this model would fail. In addition, another constraint would be the high ratio of students to the teachers, which has limited the learning efficiency for each individual learner [3]. Thus, the traditional education model "lecture-and-book" is becoming more and more inefficient. Research has shown that non-linear knowledge structure is superior to the traditional linear text-based structure in terms of knowledge diffusion [4]. Therefore, distance education must exploit new models to deliver knowledge to the learners, so it is shifting towards non-traditional learning styles where courses are provided through TV broadcasting, CD-ROM and Telelearning Systems.

In this section, a brief overview about the most important non-traditional learning models is to be given. These models are classified into Broadcasting model, CD-ROM Courseware model and Telelearning model, based on the utilized infrastructure, provided services and level of scalability. The utilized infrastructure encompasses the employed hardware and software within each distance education model. The provided services indicate what kind of learning services would be offered to the learners by such a model,

e.g. the interaction between students and teacher. The level of scalability is determined by number of students that could use the system locally or remotely. We have arranged those models starting from the simple one “Broadcasting model” to the most complex one “Telelearning model”. Each one of these models has its’ own advantages and disadvantages.

### **Broadcasting Model**

Broadcasting model is the first form of distance education systems [1]. It requires less of an infrastructure of hardware and software. For instance, once you have a TV at home or in your office, you can easily attend a course by watching the broadcasting lectures [3]. Some Broadcasting systems have utilized video conferencing technology to offer lectures to distant learners. However, these systems are affected by some constraints. For instance, learners have to follow the time schedule of the broadcasting center and cannot take the class at the most convenient time for them [3]. In addition, learners could not interact with their teacher(s) nor control the learning process according to their own abilities and cognitive capabilities. It is a passive learning manner, i.e. it is a one-way learning process from the teacher to the students without any sort of interaction. Although, some enhanced broadcasting systems have tried to solve this problem by adding the telephone service for real-time communication between students and teachers, it can provide only a very low level of interaction.

### **CD-ROM Courseware Model**

Some researchers have called this model a “PC Model” [3], but the utilization of computers is not restricted to this model only. The CD-ROM Courseware model has improved the Broadcasting model in terms of quality and quantity. It has exploited the

achievements within multimedia information processing and CD-ROM technologies to deliver and render a full multimedia courseware presentation. Courses can be stored and delivered by Compact Disks (CDs) and finally presented on computers. Courseware materials may comprise a lot of images, graphics, animation, voice and even audio/video clips in order to enrich the courseware presentation. This model has offered several advantages, such as courseware material being made available for access by a learner at any time, thus, the learning speed is controlled by the learner himself. In addition, a courseware on a CD-ROM provides some sort of interaction between the learner and courseware presentation agents “*GUIs*” to facilitate the learning process.

On the other hand, this model has some constraints other than time and space. First, since the storage capacity of the CD-ROM is limited, so the quantity of knowledge to be transmitted would be reduced. Second, knowledge delivered by a CD-ROM is static [3][4], because most of the CDs are of the type “write-once-read-many”, so it is difficult to update courseware content or to replace a part of its out-of-date material with new. The most suitable way to update the CD’s content is to throw away the old one, and order a new one, which is cost effective. Furthermore, this model did not provide students-to-teacher or students-to-students interaction. Based on this model, the phenomenon of educational groups cannot be supported. Therefore, a new model is proposed and designed to overcome all these constraints and offer an enhanced distance education environment. A Distributed Computer-based Model “Telelearning Systems” forms a new generation of distance education models. We believe that Telelearning systems must be designed in a way that supports the learning services, facilitates the education process and achieves the education goals.

## **Distributed Computer-Based Model**

As mentioned earlier, the previous distance education models are affected by several constraints that would create many obstacles for distant learners. Although, the Broadcasting systems and CD-ROM Courseware systems are the most commonly used systems, they do not support scalability, full accessibility and two-way interactivity.

With the emerging infrastructure and enhanced multimedia information processing technologies, innovative Telelearning Systems can be designed and implemented to deliver courseware content locally or remotely to distant learners. Telelearning Systems have offered several features to build a seamless education environment. First, these systems aim at making the education process more accessible and reachable by large groups of users called educational groups. This provides more opportunities for those who could hardly access education resources through the traditional learning style [2]. Second, because of the achievement within multimedia computing and networking infrastructure, computer-based learning becomes more powerful by integrating multiple media of information into the system and providing more effective and expressive ways to represent knowledge. Third, Telelearning Systems try to present knowledge structure in a form closer to the reality, by using sophisticated multimedia presentation environments. In addition, students are offered a real-time, interactive and reusable information interchange through different platforms. For instance, a student can be easily connected to a teacher or an on-line facilitator to obtain help on a specific topic using e-mail, chat, telephone connections and video conferencing. Furthermore, these systems also provide the ability to come in contact with other students from different social, cultural, economic and experiential backgrounds to discuss topics

of similar interests. In other words, sophisticated Telelearning systems involve interactivity between teacher and students, between students and the learning environment, and among students themselves, thus resembling active learning in the classroom [1][5]. As a result, students gain not only new knowledge but also new social skills, including the ability to communicate, interact and collaborate with widely dispersed colleagues whom they may never have seen [1]. Interactivity represents the connectivity the students feel with the distant teacher, their colleagues and on-line facilitators. Without such connectivity, distance education degenerates into the old correspondence course model of independent study, where students become isolated and eventually drop out. However, there is still a considerable lack of dialogue in Telelearning systems when compared with face-to-face classes [1].

Nevertheless, Telelearning systems are being restricted by networking capabilities and information coding methods [3]. Meanwhile, the World Wide Web (WWW), in particular, opens wide the door for offering courses to distant learners. What makes the WWW attractive to an educational institute, is a large communication network to exchange information in two ways; namely, the on-line browser and the courseware package distribution [6]. In spite of the WWW benefits, it is still a very slow communication medium. It frustrates students who are accessing outside the university's campus, because the current data transfer rates are very low.

Currently, several practical and experimental forms of Telelearning systems are being developed and placed on the WWW. For example, United Kingdom's Open University, Vancouver's Open Learning Agency, K-12 programs, Project BIO [7],

Norway's NKS Distance Education System, etc [1], as well as our Telelearning System "MITS", which will be described within the next chapters.

### **1.1.3 Distance Education Participants**

In the traditional education environment, teachers interact directly with their students, while, in distance learning, the relationship is not only between teacher and students. A lot of participants are involved in distance education environment; for example, distance education systems' designers, courseware authors, teachers, on-line facilitators, producers, media specialists, database administrators, service providers, technicians and, of course, students. Each participant has specific roles to play in order to integrate the distance education environment.

Some participants' responsibilities have been changed from the ones they were used to doing in traditional education, to new ones in distance education. In this section, the most important participants will be addressed, along with their new roles, while others will be discussed within the next chapters.

#### **Distance Education Designers**

Distance education designers have to consider a variety of issues that could affect the distance education process. The most important issues that need to be considered are: the learner's characteristics and needs, the influence of media upon the education process, strategies to increase interactivity as well as active learning, accessibility, and the new roles of teacher, on-line facilitator and student. Although, technology is an integral part of distance education, any successful distance education system must focus on the instructional needs of the students, rather than on the technology itself [1]. It is also

essential to consider students' interests and experience, educational levels, instructional problems and familiarity with Telelearning methods and courseware delivery. Thus, designers must build distance education systems that should stand on solid ground.

Intelligently designed Telelearning system can make a significant and positive difference in the way students are learning.

### **Teacher**

In a conventional education environment, teachers interact directly with their students. It is face-to-face or one-on-one communication. Teachers prepare their supported materials, lecture notes, assignments and tests. However, the distant teacher's role has been changed from the controller of a classroom to a consultant who could help students. In distance education, teachers have to be allowed to choose, willing to make choices and qualified to implement their choices effectively [1]. In other words, flexibility should be granted to teachers to develop their personal teaching approach utilizing the variety of options offered by technology [8]. In addition, distant teachers are not in direct classroom contact with their students. Communication is provided through videoconferencing, chat or e-mail and is mediated by a host of team partners, which may include editors, designers, producers, technicians, media specialists, site facilitators and service providers [1]. Therefore, it is essential to prepare a well-set plan and coordinate staff's activities to construct a courseware as well as provide enhanced interaction mechanisms in order to deliver courseware content to distant learners.



## **On-line Facilitator**

The on-line facilitator is an extension of the site teacher, though he/she need not be a teacher [1]. The on-line facilitator does monitoring and tutoring of students via e-mail, chat, telephone or video conferencing facilities, as well as providing help when a student encounters a problem during the learning process.

## **Courseware Author**

Courseware author is responsible for constructing a courseware. It is possible that there is not just one author but many authors for the courseware, including multiple secondary-authors. Courseware authoring is based on a courseware data model, which is generated according to multimedia document model [9][10] and teaching architectures [5]. The proposed courseware data model will be addressed in chapter 3.

## **Distant Learners**

Distant learners are students who access courseware content and being able to work on their own, they are the center of the learning process. Moreover, they have additional duties to do, e.g. they must learn how to discriminate between “junk” information and “quality” information, to distinguish facts from persuasion, and to understand how the technology itself shapes the information it carries [1]. In other words, distant learners must analyze the information that they are reading, listening and viewing, and then contribute their analysis to distant teachers and/or on-line facilitators. Such analysis and hard work provide more opportunities for students to understand the courseware content and improve their learning process.

## 1.2 Terminology Related to Thesis Work

In this section, some important terms that are used in the thesis are going to be introduced for the convenience of reading.

- **Multimedia:** A general term indicating the merging of three industries: computing, communication and broadcasting [11].
- **Multimedia system:** A software application that supports the integrated processing of multiple media types such as video, audio, image, text, animation, voice and graphics with at least one time-dependent medium [11].
- **Multimedia database:** A large collection of media data objects on secondary storage, associated with a set of programs and operations used to manage, manipulate and maintain the information in the database [12][13].
- **Distance education:** It puts emphasis on the separation of the learner and the educator in space and/or time [1].
- **Learning:** It refers to “a relatively permanent change in behavior or knowledge, brought about by practice or experience”. A piece of information is said to be learned and becomes one's knowledge when it is understood and memorized [4].
- **TeleLearning:** A learning style that ensures “knowledge diffusion” to distant learners utilizing distributed systems.
- **Courseware:** An educational software entity that contains different knowledge components, yet it resembles the objectives of a traditional course.
- **Distant learner:** A student who could use a distance education system to access courseware content.

- **On-line facilitator:** An educator who provides help when a student encounters a problem during the learning process.
- **Courseware authoring:** The construction of a courseware that involves choosing the media objects, applying a teaching architecture, specifying the scenario, and so on [2][3].
- **Multimedia document:** A single entity that provides an integrated and homogeneous way to describe, organize and structure multimedia information objects and to represent their temporal relationships [9][10].
- **Database schema:** The utilized data structures associated with a database.
- **Persistency:** The ability to store data permanently in a secondary storage [14]. In other words, it is the ability of data objects to survive through different transactions and program invocations [12].
- **Persistent-capable class:** The capacity of class's instances to be stored in a database.
- **Persistent-aware class:** A class can manipulate persistent objects, but cannot itself have instances stored in the database [15].
- **Annotation Process:** A database management system API "Class File Postprocessor" automatically inserts the required code into application classes, in order to be persistent in the database [15].
- **Linear knowledge structure:** Knowledge that is represented linearly for the learners to perceive. For example, knowledge in books is structured linearly in content, and it is assumed that learners should perceive them page by page [3].
- **Non-linear knowledge structure:** Knowledge that is represented in a cross-reference manner. In this structure, learners can perceive knowledge non-linearly, following the

links between related nodes instead of layout pages. Non-linear knowledge structure is closer to the real world knowledge than linear structure [3].

### **1.3 Thesis Objective**

The main objective of the thesis is to design and develop a Telelearning System that is intended to be used in a distance learning environment. Specifically, emphasis will be placed on designing a courseware data model, modeling the database schema and developing database engine modules as well as courseware presentation agents. These aspects are considered as the backbone of the system.

Upon completion of the design and development phases, an integrated system called “MITS” provides an integral view to the distance education era. The system is developed to deliver courseware content to distant learners over the Internet

### **1.4 Thesis Outline**

The main contributions of the thesis are the proposed courseware data model, the design and implementation of the database schema, the development of the application-server modules and samples of the results.

The remaining of the thesis work is organized according to the following outline. Chapter 2 introduces a Multimedia Interactive Telelearning System “MITS”, which is developed at the Multimedia Information Research Laboratory (MIRL) at the University of Ottawa. It presents MITS objectives and requirements. Then, it focuses on the system architecture and its major components. It also provides a brief overview of the previous approach of MITS, which is based on MHEG technology and proposed by Wang [3].

Finally, it illustrates the drawbacks of such MHEG-based model for MITS. Chapter 3 describes the design of a courseware data model. An overview of the “multimedia document model” and “teaching architectures” is presented. The object-oriented paradigm and multimedia document model were employed in order to facilitate the design of the courseware data model. Then, the generated courseware data model is presented. Chapter 4 is considered to be one of the major contributing parts of the thesis. It describes MITS database management issues. First, a brief overview of a multimedia information system called MEDIABASE will be given. Second, the employed database tools “ObjectStore environment” will be presented. Then, MITS database schema is described in major detail. It also presents MITS database management architecture based on client-server computing. Finally, the persistency states of data will be described from the ObjectStore point of view. Chapter 5 describes the implementation issues of the system. It will address, briefly, the exploited programming environment “Java environment”. Then, the implementation of the database schema and application-server modules will be described. In addition, the postprocessing or annotation of MITS classes will be discussed. Sample results of the constructed courses will be presented. Finally, the system integration will be addressed. Chapter 6 summarizes the thesis work and provides suggestions for the future research.

# Chapter 2

## 2 Multimedia Interactive Telelearning System (MITS)

This chapter describes our distance education system from several perspectives. First, it presents the system's objectives and requirements. Second, it focuses on the system architecture and its major components. Finally, it provides a brief overview of the previous version of the system, which is based on less efficient technology - "MHEG technology". It presents briefly MHEG-based model that was proposed previously for the system, and it also addresses the drawbacks of such model.

Our new approach of the system is entirely based on the Object-Oriented paradigm, new trends at multimedia computing and utilization of advanced technologies. It is an innovative Telelearning system, which utilizes an object-oriented database for supporting database capabilities, Java environment for developing the system's software modules, the Internet for providing the communication network, enhanced techniques for capturing as well as coding multimedia information, etc. We have proposed a courseware data model that describes and represents courseware content in order to facilitate courseware authoring and ensure its delivery to end-users.

### 2.1 Objectives and Requirements

The Internet is an exciting new medium for providing courses to distant learners. It is considered an ideal vehicle for effective courseware delivery to users anywhere in the world at any time [16]. At our laboratory at the University of Ottawa, we are

developing a Multimedia Interactive Telelearning System (MITS). The goal of the system is to build an environment for delivery of courses over the Internet. Our approach represents a significant advance over the typical Internet approach of delivering educational information using text, static images and a few video clips over hypertext HTML pages. The new approach aims to deliver courseware elements “objects” to distributed users smoothly. The strength of the system stems from the proposed courseware data model as well as database schema, the developed database engine, the integrated system’s software modules, the generated courseware presentation agents and the courseware reusability. All these issues will be addressed within the next chapters.

Several aspects have been taken into consideration during the MITS’s design stages. First, the system should enable students to seek courses-on-demand, which offer students the flexibility to access multiple scattered database servers from any access point in a network at a convenient time. Second, the learning process may be controlled according to the students’ learning styles and cognitive capabilities [2][5]. Third, the system should provide interaction mechanisms and incorporate a number of multimedia features, which can give the learning environment a new and refreshing flavor [17]. Fourth, the role of an instructor has been changed from that of controller of a lecture to the courseware author and/or on-line facilitator, who is responsible for providing help when a student encounters a problem during the learning process. Thus, a student should be easily connected to a teacher, or an on-line facilitator to obtain help on a specific topic using e-mail, chat services, telephone connections and video conferencing. Furthermore, the system should be capable of handling various student profiles associated with different learning objectives. It is therefore important that the system should be

developed for different courses and/or students demands [2]. In addition, our challenge is to create an enhanced setting that will be a seamless education environment that will encourage reflective practice among students and teachers [16].

Briefly, MITS encompasses different major components: courseware authoring site, courseware database server, multimedia database server, media production center, courseware presentation agents “Rendering application and GUIs” and an on-line facilitator (Figure 2.1). A detailed system architecture is illustrated in the next section.

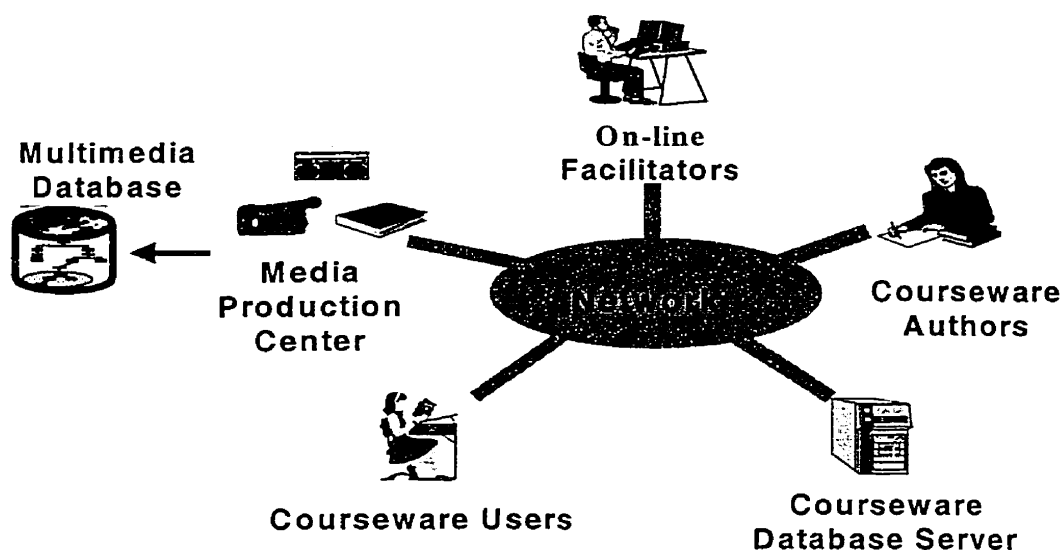


Figure 2.1 MITS's major components

## 2.2 Generic System Architecture

Our system “MITS” is a distributed Telelearning system that satisfies the previously mentioned requirements. MITS is based on the client-server architecture. It is composed of several basic components (Figure 2.2), where each one of them has specific roles to perform. These components are: courseware authoring site, courseware database server, multimedia database server, database schema modules, media production center,



media managers, courseware presentation agents, video indexing techniques and real-time communications system.

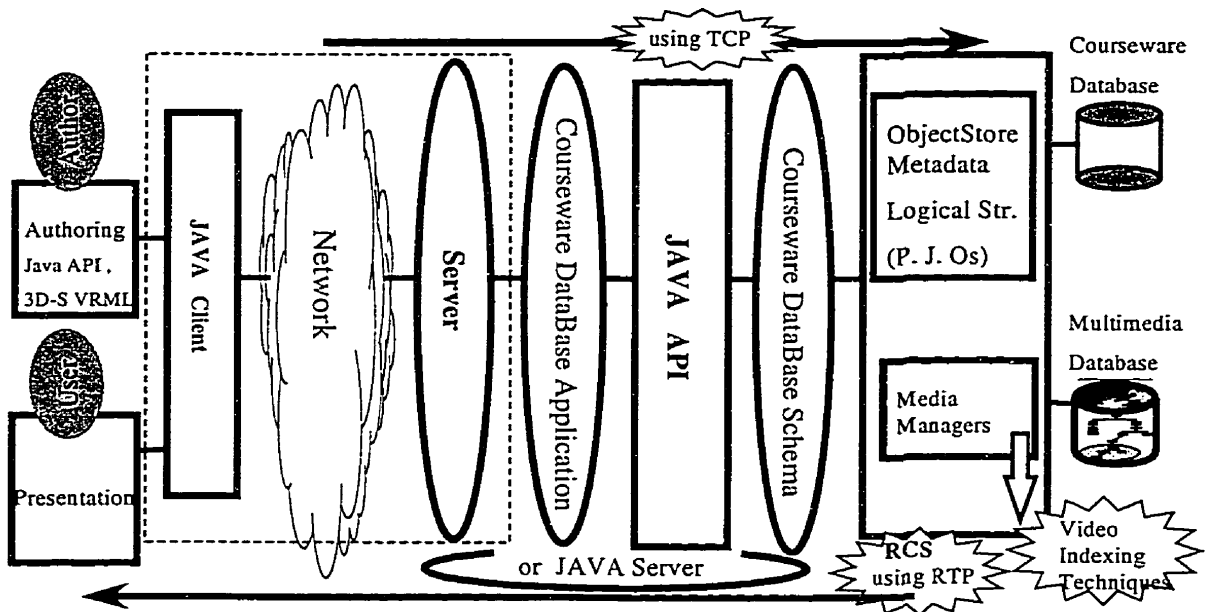


Figure 2.2 MITS general architecture

- Courseware Authoring Site** is the *factory* for constructing courses based on a courseware data model, which will be explained in chapter 3. Object-Oriented paradigm [18] and multimedia document model [9] have been exploited to design the courseware data model. The courseware author constructs a courseware within two basic steps. First, the courseware data model is employed to define the courseware hierarchy and identify the courseware presentation structure by specifying the relationships among courseware content “physical media objects”. Second, the content objects of the courseware are specified either by creating new ones through the Media Production Center, or by making reuse of those ones that are already stored in a Multimedia Database server (MM-DB server). Then, the constructed courseware is stored in the Courseware Database server. The Courseware Authoring Site exploits

Java environment and Virtual Reality Modeling Language (VRML2.0) towards the courseware creation.

- **ObjectStore** is a commercial Object-Oriented Database Management System from Object Design [19], which is used as the basis for the database capabilities. It is a product based on making C++ a database programming language. ObjectStore5.0 utilizes Java and C++ as programming environments for implementing client applications (Authoring and Presentation applications) and the application server. It provides an interesting combination of full support for database capabilities and object-oriented programming features. MITS database issues will be addressed in chapter 4.
- **Database Schema Modules** are dedicated to the management, manipulation and maintenance of database schema aspects of a courseware. Once an author constructs a courseware, the courseware meta data (i.e. the courseware logical structure and presentation structure) are transferred for storage from the client authoring site to the Courseware Database server as Java classes and VRML objects using reliable TCP. A CoursewareDB Application Module (CDBAM) is designated to include the definition of each courseware class as well as its instances, and then to annotate and mark these volatile classes automatically or manually to be *Persistent-Capable Java classes*. These classes are mapped through the Java API module, and are to be stored persistently applying a CoursewareDB Schema Module (CDBSM), which resides at the server site. CDBSM includes the definition of each persistent-capable class of objects and provides a pipe stream to store these courseware classes at the Courseware Database server. An enhanced database management architecture is

described in chapter 4, which defines MITS database modeling and schema generation that are produced by utilizing different software engineering techniques and ObjectStore. In addition, students' requests to the database are supported through a mediator application layer "Courseware Presentation Agents". This offers two advantages: it protects the courseware database from unauthorized accesses, and hides the details of the database operations. As a result, students are offered the flexibility to access information repositories. Also, databases are managed, manipulated and maintained properly.

- **Media Production Center:** is responsible for capturing information from the real-world and coding them into various types of "physical media objects", such as video, audio, image, graphics, animation, and text. These media objects will be employed as basic multimedia information for the courseware creation and presentations. Multimedia data can be categorized as static or continuous media. The term "static media" refers to the media that does not have a temporal dimension, while continuous media has an implied temporal dimension [20]. Video and audio are the best examples of continuous media. The rendering of video must satisfy strict temporal constraints and must also be synchronized with the associated audio [20]. In addition, the media production center normally contains different equipment for capturing media information, such as video cameras, VCRs, scanners, speakers, microphones, and so on. McNabb [1] noted that more experimental studies are needed in the area of media selection, which is one of the basic steps towards the courseware authoring.
- **Multimedia Database Server:** the multimedia information that is already captured from the real-world and coded into different types of media objects are indexed by

textual references, then stored at a Multimedia Database server (MM-DB server) by exploiting a set of Media Managers. Meanwhile, the courseware meta data includes these textual references as pointers that facilitate the retrieval of those media objects at rendering time is stored at Courseware Database server. One of the most important issues related to media objects is the storage. Media objects, especially video data, can be very large in terms of bytes. However, there are several compression standards that have been designed, implemented and utilized in order to manage, compress and store those media objects at a MM-DB server.

- **Media Managers** are part of ObjectStore tools. These managers are a set of class libraries used to facilitate the storage and retrieval of those media objects into a MM-DB server. A variety of media managers are provided within the ObjectStore environment [15]. For instance, Text object-manager, Image object-manager, Video object-manager, Audio object-manager, HTML object-manager, etc.
- **Courseware Presentation Agents** are rendering client applications, which are responsible for offering distant learners a variety of multimedia interactive courseware services through sophisticated Graphical User Interfaces “Rendering GUIs”. These services enable students to browse all courses that are stored in the Courseware Database server, or to retrieve the logical structure of a specific courseware. It is then possible to retrieve a specific section from within such courseware, enclosed with the pre-defined presentation structure by playing back its basic media objects. The rendering application modules interact with students through standard browsers such as Netscape3.0 [21] or Internet Explorer4.0 [22] in order to ensure the courseware delivery. When a student encounters a problem during the

learning process, he/she can always get help from the on-line facilitator through e-mail, chat, telephone or video conferencing facilities. Therefore, the power of multimedia in supporting the learning process is not only in the ability to combine text, audio and visual data, but it is also evident when combined with rendering applications to provide interactive functionality for users to navigate for information at their own pace [4]. The integration of the database engine with the courseware presentation agents will be addressed briefly at chapter 5.

- **Video Indexing techniques:** MITS is associated with ObjectStore to provide a high level of indexing techniques based on the content of such media objects as image content or video frames content. Currently, ObjectStore only offers image-indexing techniques, such as content-based retrieval [15], which is based on image visual properties, including color histogram, texture, shape of objects and sketch. Deriving such features requires automatic analysis of the multimedia information. The primary methods used for image data are image processing and image understanding [12], while the primary methods utilized for video data are video shots analysis, video parsing and video shots abstractions (i.e. selecting key frames to represent each shot) [23].
- **Real-time Communications System (RCS)** is a part of MITS system, which was developed at our Laboratory [24]. It was designed to experiment with media-on-demand issues using Real-time Transport Protocol (RTP) over the network with emphasis on video and audio media types. The main features of the RCS sub-system are real-time transmission, scalability, utilization of standard protocols, flexibility and user interactivity [24]. RTP was used over UDP for real-time transfer of video/audio,

rather than reliable TCP, because when packet losses occurred, retransmission and congestion control methods used in TCP resulted in gaps during media presentation.

All these components are distributed over the Internet to provide real-time interactive multimedia courseware services to a variety of educational groups in order to satisfy their learning needs.

## **2.3 MITS Previous Approach**

A previous approach of MITS was proposed and designed at our Laboratory [3]. It is entirely based on MHEG technology. Briefly, the MHEG standard will be described, and the proposed MHEG-based model for MITS will be introduced, followed by the illustration of the drawbacks of such model. However, our new approach of MITS does not use MHEG technology; it is designed based on the courseware data model.

### **MHEG Technology**

MHEG stands for Multimedia and Hypermedia Information Coding Expert Group. The standard it provides is Coded Representation of Multimedia and Hypermedia Information, which is commonly called MHEG standard [25]. MHEG is a developing international standard that is providing a coded representation for multimedia/hypermedia information to be used, and interchanged in real-time, by applications in a wide range of areas and on heterogeneous platforms [3][9]. For instance, interactive multimedia applications and document interchange services.

The MHEG standard is based on the object-oriented notations, but it does not need an object-oriented system to be implemented [25]. MHEG technology aims to

define a “*Framework*” for several multimedia and hypermedia applications. This “*Framework*” comprises the coded representation of independent and elementary units of information, which will be specified as “*Objects*”, and utilized or interchanged by different applications. In addition, the standard aims at sustaining real-time interchange and presentation using minimal resources [3].

Research on MHEG began in the late 1980’s, and the standard is developed in a number of parts [25]. For instance, Part V of the standard is developed to support the distribution of interactive multimedia applications, based on the client-server architecture across heterogeneous platforms. However, these parts are out of the scope of our thesis work.

Basically, MHEG standard contains eight types of classes. These classes are MH-Object class, Composite class, Content class as well as Multiplexed Content class, Script class, Action class, Link class, Container class and Descriptor class [25]. In addition, system designers are able to add new classes into the system. Based on these classes, MHEG objects can be instantiated by the object designers and interchanged between the applications. Messages for the communication between MHEG objects, or for the communication of these objects with client applications, are specified in the MHEG standard. However, it is up to the applications or services to define the way to utilize or handle them, and if necessary encode them [3].

### **MITS Based on MHEG Model for Information Interchange**

Wang [2] had proposed an MHEG model for interactive multimedia courseware delivery system (Figure 2.3). It is a layered model, which contains multiple layers, such

as application layer, script layer, MHEG object layer, non-MHEG content object layer and the communication protocol layer. All these layers are located in three sites: authoring, storage and presentation. These sites are the key factors for the process of transmitting knowledge to distant learners.

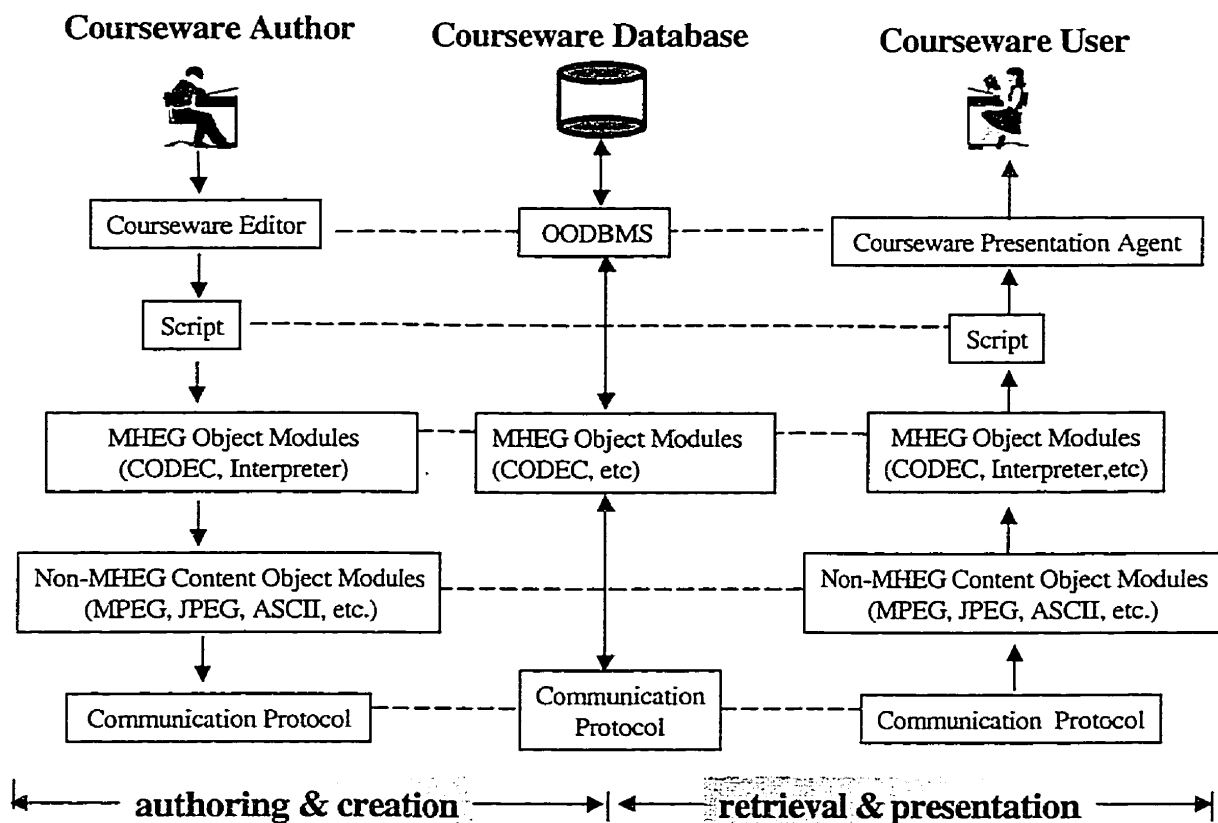


Figure 2.3 MHEG-based model for MITS

At the application layer, the authoring site utilizes a courseware editor, the storage site exploits a courseware database, and the presentation site uses a courseware navigator. The second layer of the model is the script layer, which is employed to specify complex relationships between MHEG objects and run-time objects for the courseware presentation [2][3]. The next layer of the model is the MHEG object layer, where, at the authoring site, objects are coded into ASN.1 or SGML format using *MHEG Engine* and



transmitted through the network. When the objects are received at the presentation site by the courseware navigator, they are decoded and interpreted for rendering using the *MHEG Engine*. In addition, this layer is responsible for sustaining other activities that facilitate the presentation process, e.g. resolution of object references, creation of runtime-objects, and interpretation of link as well as action objects. The *MHEG Engine* is a set of software modules designed and implemented by the system designer to encode, decode, handle or interpret the MHEG objects [3]. Moreover, the non-MHEG content object layer is responsible for offering mechanisms to handle various types of coded media objects. These content objects are captured and coded at the media production center utilizing different data coding standards, such as MPEG, RealAudio, JPEG and ASCII, depending on the media types. Finally, the communication protocol layer is dedicated for exchanging messages and acknowledgments between the applications. Several networking infrastructures can be used to offer such communication services; a broadband network had been suggested for the first approach of MITS [2].

## **2.4 Drawbacks of MHEG-based Model of MITS**

In order to use such MHEG-based model for MITS (Figure 2.3), a specific MHEG flavor application, a user interface, a MHEG engine and a communication management module are common modules to be installed at every user's site.

This spawns several obstacles that affect the first approach of MITS in terms of the scalability and performance. Since each end-user's site has to install the MHEG-based presentation application and the *MHEG Engine*, students are unable to access the

system through standard browsers such as Netscape [21] or Internet Explorer [22]. Thus, the system's scalability is limited. In addition, as the MHEG-based model has several layers, so courseware content must go through all these layers in order to be delivered to end-users. Therefore, the system's performance is slow. Moreover, MHEG technology itself is not stable, because there are different versions of the standard. Each one of them includes new specifications. This makes the implementation process of any system that utilizes MHEG technology more complicated. In addition, MHEG includes a limited synchronization specification support. It is suggested that other standards such as AudioVisual Interactive Scriptware (AVIS), can be used to handle the more complex synchronization requirements of presentation scheduling [9]. Furthermore, recent comparisons have revealed that MHEG's very high conceptual overload makes authoring MHEG documents cumbersome [26].

# Chapter 3

## 3 Description of Courseware Data Model

In chapter 1 and chapter 2, we have pointed out that the courseware author constructs different courses based on our courseware data model, which is designed according to the “multimedia document model” [9][10] and teaching architectures [5]. The Multimedia document is an architecture that aims to describe, model and structure multimedia objects, while teaching architectures incorporate the learning theory with the computer capabilities to accomplish one-on-one based teaching and satisfy student learning needs. The teaching architectures are different from traditional teaching methods that are used in typical classrooms. Most of these teaching architectures concentrate on the idea of learning by doing, which enables distance education developers to create educationally effective Telelearning systems.

Basically, the courseware data model aims at describing and representing courseware content in order to simplify the courseware creation and support the content delivery to distant learners. An overview of the multimedia document model and teaching architectures is addressed in sections 3.1 and 3.2 respectively. Then, the courseware data model is described in section 3.3.

### 3.1 Overview of Multimedia Document Architecture

The first step toward the design of a multimedia information system is to provide an integrated and homogeneous way to describe, organize and structure multimedia

information objects and to represent their temporal relationships in a single entity called the “multimedia document” [9]. Multimedia documents differ from traditional documents that are composed of text and graphics [10]. Documents that comprise a combination of different media types such as video, audio, graphics, animation and text can express or present ideas more clearly than traditional documents. Multimedia documents are originated from “*Office Document Architecture*”(ODA). The ODA [9][27] supports only static media types such as text, raster graphics and geometric graphics, while, continuous media such as video and audio cannot be incorporated into the ODA. These media differ from static media in that they are laid out over time and have temporal properties. Thus, ODA does not address the temporal relationships between media items within a document [9]. The emergence of continuous media (e.g. video and audio) imposes new requirements on document representation and information storage. Therefore, in order to include continuous media and support temporal representation; ODA must be modified and extended to a new model called the “*Multimedia Document*”. It is an architecture aiming to model multimedia objects. The new generation of multimedia documents is able to integrate new types of information, such as continuous media (e.g. video and audio) and computer-generated media (e.g. computer graphics and animation) [27]. Examples of multimedia documents are courseware, textbooks, atlases, medical reports, electronic news, and so on.

There are two types of multimedia documents: passive documents and active documents. In passive multimedia documents, the author integrates continuous media simply by representing them in a static visual form such as a frame for video and an icon for audio [9]. In active multimedia documents another approach is used to integrate

continuous media. Each media item must be treated as an *object* to be presented in time; each object is rendered for a specific duration of time. Thus, by assigning duration to every object in the document, the author can create a presentation schedule to describe when each object in the document should be presented. In contrast to the passive documents, active multimedia documents play back in a presentation that changes continuously in time. At our laboratory, a structured description of the multimedia documents called “*Mediadoc*” had been proposed [9]. This is an architecture for the creation of active multimedia documents. It also includes a rendering synchronization scheme that enables the specification of temporal characteristics for multimedia objects and relations between them.

However, two major problems that appeared with multimedia document architectures and authoring systems: are limited functionality and poor authoring environments [9]. Considering these two problems, few goals were established for the development of “*Mediadoc*”. First, the generated architecture must be powerful enough to describe multimedia documents to the extent required by authors. Specifically, it should offer a beneficial set of synchronization specification types for creating the presentation schedule of a multimedia document. Second, the architecture must be simple enough so that authors can readily create multimedia documents that are clearly and concisely structured and easily interpreted, understood and modified [9]. In addition, the procedure of creating presentation schedules for multimedia objects should be easily driven and not tedious.

The most important issues of the document architecture are the logical structure, layout structure and rendering scenario, which describe a document’s content and specify

how the content will be laid out and played back. The logical structure describes how a document is organized into major components and sub-components. Thus, a document can be structured into a number of levels. Each level will contain several sub-documents. At the lowest level, sub-documents will represent media objects [10]. The layout structure describes the spatial properties of media objects that will be presented during play back. The layout process requires assigning each piece of information to a “rendering area” on a display device. In addition, the layout process will be completed once the document’s rendering schedule “Scenario” has been specified. A scenario is defined as a schedule for document play back. These scenarios describe when, and for how long, each media object will be rendered. In addition, such scenarios specify the temporal synchronization that coordinates the real-time presentation of a multimedia document, and maintains the temporal ordering (i.e. time-ordered relations) among the media objects [10]. In other words, a “scenario” makes it possible to schedule multimedia events to happen according to specific relationships between media objects. Therefore, scenarios are essential for multimedia document play back because they provide the means of integrating static and continuous media. Each multimedia document can have several scenarios representing different ways it can be rendered. More details of “Mediadoc” architecture are provided in [9].

## **3.2 Teaching Architectures**

Regardless of the traditional teaching methods, Schank [5] had proposed a number of teaching architectures that incorporated the learning theory with the computer capabilities to accomplish one-on-one based teaching and satisfy students’ learning

needs. These teaching architectures include simulation-based learning by doing, incidental learning, learning by reflection, learning by exploring, case-based teaching and goal-directed learning. The research and studies throughout the learning theory have demonstrated that immediate and frequent feedback, cooperative learning, and well structured exposition of information and data can improve the learning process [16].

### **Simulation-based Learning by Doing**

This architecture is usually used when apprenticeship will not work all that easily or if it is risky. In other words, simulation applications allow users or students to experience with difficult events or tasks and try to gain experience from them, as for example in pilot training. It is composed of four parts: a student, a simulator program, a storytelling program and a language understanding program. A student can receive training through the simulator program, where a language understanding program will interpret student questions to languages that the computer can understand. A storytelling program is activated by a trainer (i.e. teacher) at appropriate times to tell stories from the experiences of experts in actual situations [3][5].

### **Incidental Learning**

This architecture creates tasks whose end results are inherently interesting and can be used to offer significant amounts of information. The basic principle of this architecture is that students can learn easily when doing something fun, e.g. students can learn geography by utilizing a software called “Road Trip” [5]. It teaches geography to school students by letting them take simulated car trips around the United States. Upon

arrival at a destination, the student can watch exciting video clips spotlighting activities or events in that location [5].

### **Learning by Reflection**

Sometimes a student does not need to be told something, but rather needs to know how to ask the right questions. A student can be the best teacher of himself if he has someone around to listen to the ideas that he generates. In this architecture, the teacher's role is to help the students see shortcomings in thinking, and encourage them to speculate, imagine and create [5].

### **Learning by Exploring**

The learning by exploring architecture provides answers to a student's questions at the time they arise. In this architecture, students must be provided the flexibility to select their own choices and they should have a number of experts available to answer their questions. Proper organization of expert testimony is of vital importance for learning by exploring [5].

### **Case-based Teaching**

This architecture depends on two ideas: experts are repositories of cases, and good teachers are good storytellers. The students are told exactly what they need to know, when they need to know it. When students realize that they need information to progress, they will learn fast. This architecture can be combined with the simulation-based learning-by-doing architecture. The learning-by-doing architecture provides the activity, and the case-based teaching architecture provides the instruction [5].



## **Goal-directed Learning**

In order to leverage the power of the teaching architectures, we need to provide goals that students will adopt willingly. Also, we need to provide a way for students to control the environment in which they learn, and give them an opportunity to adapt what is presented to them, to their existing learning needs. Schank [5] had proposed several principles about how to build educational environments in schools and in the workplace. For instance, learning should concentrate on a task that requires transformation of skills and knowledge diffusion. The task should be challenging, but within a student's ability. In addition, an instructional designer's job is to make the learning process more attractive, which means that students will enjoy what they are doing.

## **3.3 Courseware Data Model**

The design of an appropriate data model for a multimedia information system will ensure smooth navigation and fast access between real-world application entities and multimedia information objects.

In MITS, we have adopted the multimedia document architecture "Mediadoc" [9] and object-oriented paradigm to design our Courseware Data Model (CDM). The Courseware Data Model describes and represents courseware content by specifying the logical structure and presentation information, in order to facilitate courseware authoring and ensure its delivery to distributed educational groups. In other words, CDM aims to simplify the creation of courses and facilitate their manipulation into database systems. In addition, the Courseware Data Model incorporates a courseware with a teaching

architecture, and guides students through the learning process by specifying the presentation information of the courseware content. We have utilized Java environment as well as VRML tools [28] to accomplish the design and implementation of the courseware data model.

The Courseware Data Model is composed of logical, spatial, temporal and behavioral structures. The logical structure describes how a courseware is organized into major components and sub-components, as regards chapters and sections, respectively. The spatial structure describes the layout properties of media objects that will be presented during play back. It specifies the physical location of each media object on the rendering terminal, in terms of the horizontal-vertical coordinates and the width-height measurements [10]. There is a connection between the logical and spatial structures: the spatial structure facilitates the understanding of the logical structure through typographic effects [29]. The temporal structure defines when, and for how long, each media object will be rendered, e.g. displaying a video clip for 60 seconds after a text object has been displayed. The behavioral structure is used to describe how a courseware should react to user's interactions [28], e.g. a student should click on a video object in order to be rendered.

In our Courseware Data Model, as illustrated in figure 3.1, the courseware logical structure is organized into several *chapters* where each chapter consists of multiple *sections*, although sometimes a chapter has no section. The sections' level is the leaf level of the logical structure hierarchy. While, the presentation information (spatial, temporal and behavioral structures) are wrapped in one 3D visualization container called 3D-Scenario Structure (3D-S), which is constructed by applying VRML2.0 authoring tools

[28]. Each section in the courseware logical structure is associated with one 3D-S. Even when a chapter has no section, it can contain a link to a 3D-S (Figure 3.1). A 3D-S contains multiple 3D-S Components, where each one of them represents a specific media object, such as video, audio, image and text, within a section or chapter. The spatial information of a media object is represented on X-Y plane by assigning a “rendering area” that has width and height values. This specifies the physical location of each media object. The temporal information of a media object is represented by the Z-axis, which represents the time-axis; it contains the start time of rendering the media object, the end time of rendering the media object and the length of the time it will last. The behavioral information of a media object is represented by 3D behavior arrows [28], which represent the behavioral relationship and rendering policy of a specific media object. For instance, these arrows will determine how a media object K will be rendered before/after another media object M, and how it will respond to student interaction such as clicking.

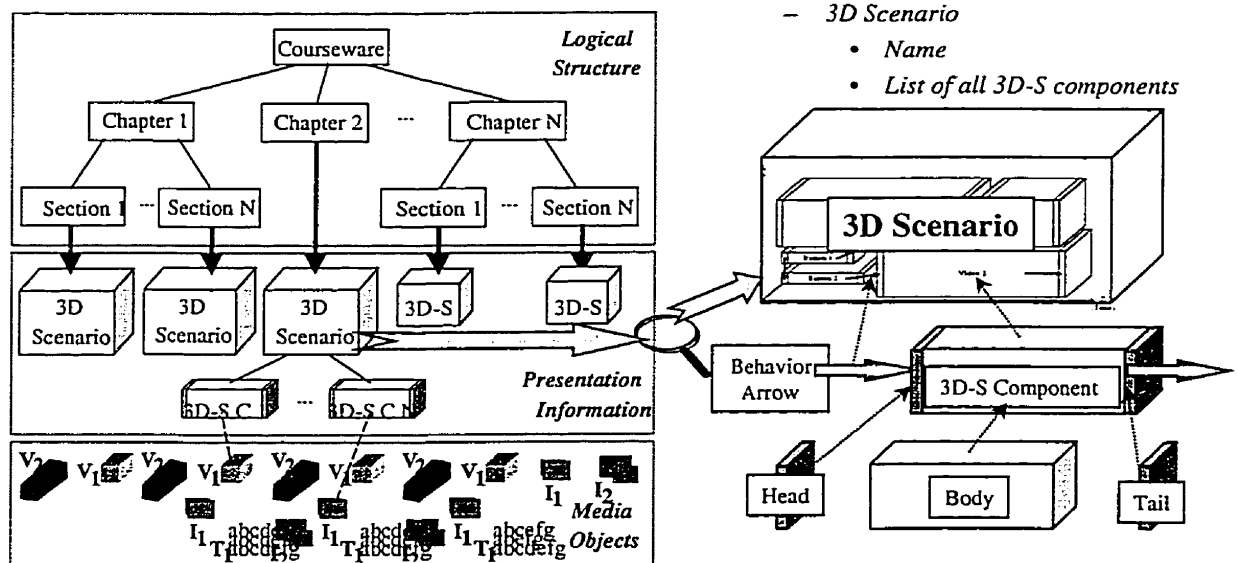


Figure 3.1. Courseware Data Model

Since the CDM is designed based on object-oriented paradigm, it satisfies different object-oriented features. For instance, it supports the “*reuse*” of logical structures and media objects. The courseware logical structure can be reused to create a new courseware by using the same chapters and/or sections in constructing different courses, e.g. section k can be copied in a courseware entitled “Object-Oriented Programming” and in another courseware called “Java Networking”. In addition, media objects can be reused within different 3D-Scenarios in the same courseware or even in different courses. In the next chapter, we will address the issue of how to translate the courseware data model to be mapped in the MITS database schema.

# Chapter 4

## 4 MITS Database Management Issues

Database management is the *backbone* of any complex computer-based system that is correlated with large repositories of information. The aim of database management is to ensure efficient storage, effective manipulation, fast access and beneficial querying to database users. Database schema is considered as the *heart* or *kernel* of that backbone. Therefore, powerful software engineering techniques must be utilized to design and generate the database schema. In addition, an appropriate database management system must be used to support the required capabilities.

In this chapter, we will explore the MITS database management issues. First, a brief overview of a multimedia information and communication system called MEDIABASE will be addressed, where our Telelearning system stems from this platform. Second, the functionality of the multimedia database management system is described by focusing on the exploited object-oriented DBMS. In addition, major contributions are presented in sections 4.3 and 4.4. MITS database modeling and schema generations are accomplished by utilizing powerful software engineering techniques. For instance, Object Modeling Technique [18] and Use Cases [38] have been employed to facilitate the schema generation. Then, MITS database management architecture is described based on client-server computing. Finally, the persistency states of objects inside the database are presented briefly.

## 4.1 Overview of MEDIABASE System

At our Laboratory, a research project called MEDIABASE that combines key aspects of multimedia, telecommunications and information processing [29]. The MEDIABASE platform is a multimedia information and communication system that has been under development for several years. It has focused on document architectures, database models, high-level communications as well as synchronization protocols, and real-time physical storage of multimedia data [9][29]. It has provided the basis for developing fully distributed and complex applications such as remote delivery of video entertainment services, real-estate information system, audiovisual interactive applications, etc [29]. An interactive multimedia newspaper application has been implemented based on the MEDIABASE platform [30]. Our Telelearning system is also one of the applications that stem from the MEDIABASE project.

The main components of the MEDIABASE platform include Mediadoc architecture, production information servers (e.g. video, image, text as well as graphics, and voice servers), communication servers (e.g. cooperative, mail and directory servers), database server, high-speed network infrastructure and multimedia user interfaces. The database server provides a set of features used to store, retrieve and manipulate multimedia documents as a whole, independent of their physical storage [29]. All these components are distributed over the OCRInet – an R & D ATM network in the Ottawa region [30].

The first prototype of our multimedia Telelearning system was implemented on the MEDIABASE system. It aimed to accomplish the implementation of a courseware

delivery system by sustaining specific tasks in the next versions of the system [3]. Thus, in our current approach of MITS, we have fulfilled those tasks through the following:

- A courseware database engine, which stores courseware logical structure, presentation information and content material.
- A client module(s) that enables users to access, manipulate and update the database.
- Courseware presentation agents that offer a user-friendly graphical interface for the learning environment [31].

## **4.2 Object-Oriented Database Management System**

Multimedia information systems deal with huge amounts of data that should be stored in large repositories of information. Thus, a powerful database management system is required to provide database capabilities. Traditionally, a database consists of a controlled collection of data related to a given entity, while a database management system (DBMS) is a collection of interrelated data with the set of programs and operations used to define, create, store, access, manage, query and present the information in the database. The functions of a multimedia DBMS basically resemble those of a traditional DBMS [12][32]. However, the nature of multimedia information makes new demands, including determining what is needed and how to provide that functionality. A multimedia DBMS provides support for multimedia data types, plus facilities for the creation, storage, access, query, and control of the multimedia database [12]. In addition, it is the task of the multimedia DBMS to provide format independence

to the applications, i.e. to supply each of the formats it needs, while hiding the internal storage formats that are actually used [13]. For the multimedia DBMS to serve its expected purpose, it must meet certain special requirements. These requirements are divided into the following categories [12][33]:

- Traditional DBMS capabilities
- Huge capacity storage management
- Information retrieval capabilities
- Media integration, composition and presentation
- Multimedia query support
- Multimedia interface and interactivity
- Performance

Traditionally, a multimedia DBMS is designed by developing a multimedia presentation layer on top of a pre-existing object-oriented DBMS (which can be truly object-oriented or relational-based), such that the core of the DBMS was developed earlier, independent of the design of the multimedia presentation layer [20]. A number of challenges are faced by the database community to provide a comprehensive solution for designing and managing multimedia database systems. These include designing new data models to capture semantics for multimedia objects, storing and accessing multimedia data, providing high-level indexing techniques for images, video and audio data, version management for distributed objects, query language development for multimedia data, etc



[33]. There are three approaches which can be taken when designing a multimedia DBMS: (1) relational DBMS + object-oriented interface + multimedia interface; (2) object-oriented DBMS + multimedia interface; (3) an object-oriented multimedia DBMS [20][33][34]. Most existing multimedia DBMSs use either approaches 1 or 2. It is well known that object-oriented styles are very efficient in supporting the development of multimedia applications [20][26].

However, the selection of multimedia DBMS depends more precisely on the application and its own requirements. The emerging multimedia applications range from multimedia display, data transfer, information retrieval, to distributed multimedia collaboration. Virtually all commercial and governmental organizations are included, with applications such as desktop publishing, education, medical, weather, entertainment, military and so on [20]. Pazandak and Srivatava [35] have addressed the general requirements of multimedia applications and provided a survey for the most popular OODBMSs and multimedia DBMS products.

In MITS, we have exploited a commercial OODBMS product (ObjectStore5.0 in Java) to support the required database capabilities [15]. We have selected an OODBMS for several reasons. It offers enhanced tools to manage, manipulate and maintain complex data such as multimedia information. This is perhaps the most important reason why OODBMSs have attracted the majority of users that have been dissatisfied with traditional databases [36]. It also sustains rich modeling capabilities that can simulate real-world environments into sophisticated systems such as MITS. Moreover, object databases can offer significant reductions in the development cost and provide substantial improvements in performance for a wide range of new generation applications (e.g.

Internet applications) [19]. In other words, the characteristics of the OODBMS can have significant impact on the performance and flexibility experienced by both application developers and users [37]. Unfortunately, OODBMSs are still subject to lively development, and the numerous proposals and prototypes differ in many aspects. It is far from clear, today, which one of the proposals will finally prevail [13].

As mentioned earlier, we have utilized an OODBMS for supporting database capabilities that are needed by MITS. Object Design [15] has produced a full object-oriented database management system “ObjectStore” and light-weight database engine “PSE-Pro” to offer the required database capabilities. These two database management systems will be addressed briefly in sub-sections 4.2.1 and 4.2.2 respectively.

#### **4.2.1 ObjectStore 5.0 Environment**

ObjectStore emphasizes on client-server architecture and offers a full database support for multiple clients distributed over the network. It is intended for applications that require high performance persistent storage for large databases, multi-user concurrent access, complete DBMS capabilities and queries over large collections of objects [19]. A library of collection types is provided in order to build database structures, including vectors, hash tables, sets, bags, lists, dictionaries, etc [15][36]. It also provides a large number of users with the privileges of accessing various databases through the ObjectStore client API, which resides on each client’s machine and is associated with application server at the server side. ObjectStore architecture has 3 layers: the client application, the application server and ObjectStore Server (Figure 4.1). The client application is implemented in Java or C++ and utilizes ObjectStore client API

to dynamically map Java objects into persistent medium for storage. The application server is implemented in Java or C++, and uses ObjectStore APIs to handle schema operations and queries. It runs on top of the ObjectStore server, either in a single process or separate processes [19]. The ObjectStore server is responsible for providing full database support such as storage management, transaction management, locking, recovery, security, etc [15]. In addition, ObjectStore facilitates the creation and maintenance of multiple versions of data, and provides support for version history. It also supports object clustering, i.e. for better performance in terms of object access time, objects that are generally processed together or those that are dependent on each other can be clustered together. When related objects are clustered, they can be simultaneously retrieved from secondary storage and cached together [36].

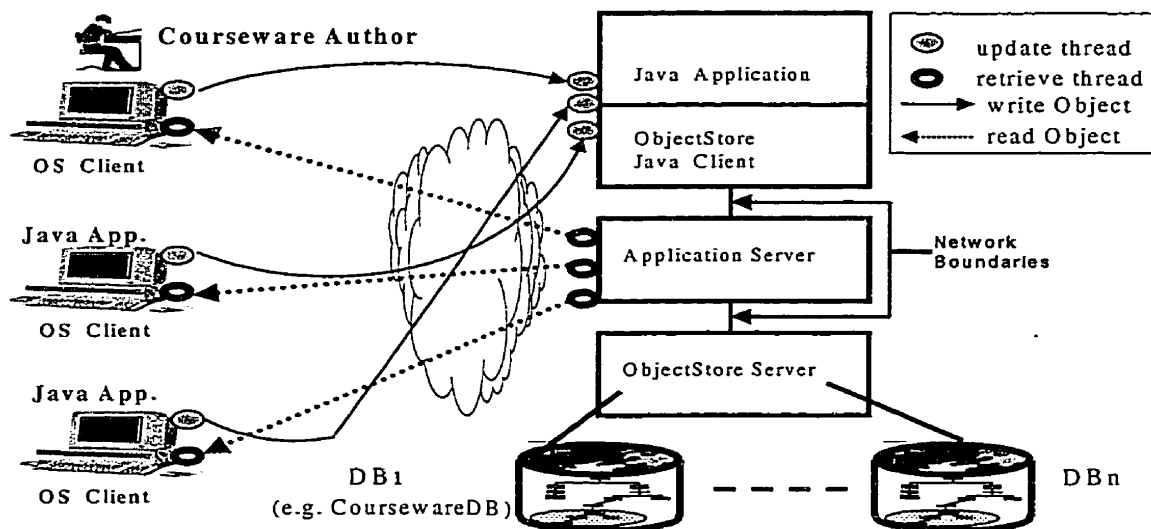


Figure 4.1 ObjectStore5.0 architecture

#### 4.2.2 PSE-Pro Environment

Moreover, a light-weight database engine that is called Persistence Storage Engine-Pro (PSE-Pro) in Java can be employed to offer database capabilities and

facilitate the generation of database schema for medium size applications (Figure 4.2). It runs separately from the ObjectStore server and is affected by several limitations [19]. It allows, at most, one user to update the database at one time, which means it locks the database at the file level once the database is under the action of an “update transaction”. But, it enables concurrent “read transactions” by utilizing multiple *retrieve threads* (Figure 4.2). PSE-Pro is not intended for a large number of concurrent users. It can support multiple readers from the database, but writers to the database are serialized since locks are held at the database level [19]. This means that a database can be updated by, at most, one application at one time. Multiple applications can read the same database simultaneously, but only one application can write to the database. It performs well for databases in the range of tens Mega Bytes. When databases start to exceed 100 Mega Bytes, PSE-Pro performance starts to degrade [19]. Therefore, in order to obtain high performance, multi-user concurrent access and indexed queries over a large collections of objects, you should consider using the full ObjectStore DBMS product.

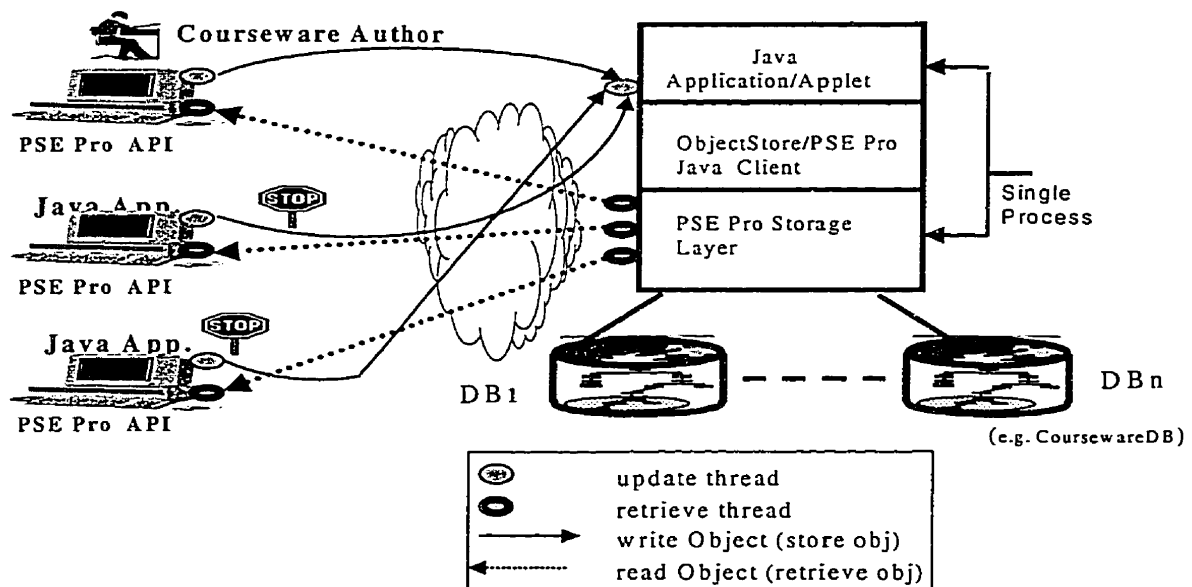


Figure 4.2 PSE-Pro architecture

### 4.2.3 Class File Postprocessor

In this sub-section, we will describe briefly an important API that is provided as a part of ObjectStore tools, which is called “*Class File Postprocessor*”. It is utilized to annotate and mark system classes (e.g. MITS database schema) to be persistence-capable classes or persistence-aware classes. Recalling from chapter 1, persistent-capable class is the capacity of class instances to be stored in a database, while persistent-aware class is a class that can manipulate persistent objects, but cannot itself have instances stored in the database [15]. In chapter 5, which describes the implementation issues, we will meet both persistence modes.

Basically, in order to store objects in the ObjectStore database, these objects must be persistent-capable. For an object to be persistent-capable, it must include code that allows persistence. ObjectStore provides the Class File Postprocessor to automatically insert the required code into application classes, which is referred to as *annotations*. Under normal circumstances, system developer(s) must postprocess together all class files in an application that he/she wants to be persistence-capable or persistence-aware. Failure to do so can result in problems that are difficult to diagnose at the application execution time [15]. Even if a particular class does not need to be persistence-capable, it is recommended that it should be postprocessed with all other class files in the application. The postprocessor has offered multiple options that allow the developer to indicate which classes must be persistence-capable, and which need to be persistence-aware, and which need not be annotated [15]. It provides a number of command options that allow developers to tailor the results to their needs.

## **Description of the Annotation Process**

- Before running the Class File Postprocessor, system developer(s) must compile all source files at the same time.
- Create a destination directory other than the source directory.
- Run the Postprocessor according to specific persistence modes.
- After running the Postprocessor, there are two versions of the application class files:
  - 1- The unannotated class files in the source directory,
  - 2- The annotated class files in the destination directory.

It is important to keep these versions separate, because when a developer runs the application, he/she must ensure that ObjectStore finds the annotated class files before it finds the unannotated class files [15]. There are several technical rules for postprocessing classes, for more details refer to [15].

## **4.3 MITS Database Modeling and Schema Generation**

One of the most important problems that the database community has tried to solve is the development of a powerful data model. Data models are essential to multimedia database systems. The data model can be used for the management of multimedia information in a way similar to the actual management of factual data by the traditional database models, such as the relational model. However, traditional data models and database systems are affected by several drawbacks for handling complex

applications such as multimedia information systems. This refers to the unique nature of multimedia information that imposes new demands, and incorporates special characteristics, such as temporal-ordering and synchronization.

Various data models such as network, relational, semantic and object-oriented models are already available for the traditional databases, and a few have been proposed for multimedia databases [12]. Some researchers have gone so far as to claim that the data model for a multimedia DBMS can only be fully achieved by object-oriented technology [12][13]. Object-Oriented paradigm aims at resolving the drawbacks and offering the flexibility to tackle the requirements of the complex systems without being restricted by the data types and query languages that exist in traditional database systems. In addition, Object-Oriented paradigm offers systems designers the power to specify the structure of complex objects and the operations that can be performed on these objects [20]. Therefore, Object-Oriented technology is suitable for building a multimedia data model (e.g. Courseware Data Model), which aims at describing and presenting data that is associated with their relationships in more complex systems such as MITS. It also enables database designers to model, create and store these complex types of data without the need for translation from complex data structures to simple table format. As a result, we have utilized object-oriented modeling techniques in designing MITS database schema.

Based on Object-Oriented paradigm, the database schema is the description of classes associated with a database. It includes all Java or C++ types of objects that have ever been stored into the database [15]. In MITS, we have adopted the Object-Oriented

model because of its richness and capabilities that support the analysis, design and implementation of MITS database schema.

### **4.3.1 Relational Modeling**

Prior to the usage of the Object-Oriented model, we mapped our Courseware Data Model to the Relational-Model and produced MITS's Entity-Relationship diagram (Figure 4.3). It facilitates the designing process of MITS database schema. First, as shown in Figure 4.3, an author creates several courses, each one of them is assigned with its attributes (title, cr\_code, keywords, creation\_date, short\_description and so on), the key attribute is cr\_code. Second, each courseware has many chapters, where a chapter is a strong entity and independent of a courseware entity. A chapter is defined by such attributes as (title, chp\_code, keywords, creation\_date, etc), the key attribute is chp\_code. Third, each chapter may contain multiple sections, where each one of them is defined by various attributes namely (title, sec\_code, keywords, creation\_date, etc), the key attribute is sec\_code. Fourth, a 3D-S entity is representing the presentation information of a chapter or a section, where each 3D-S has a name and contains a list of 3D-S Components. Fifth, 3D-S Component entity is defined by several attributes such as (name, x-y coordinates and width-height values). Sixth, each 3D-S Component is linked to only one media object, which is defined by a reference name. Seventh, all courses are stored in a coursewareDB entity, which is defined by coursewareDB\_Name. Eighth, a registered student who is associated with a profile can search or access coursewareDB for a specific courseware using a textual query. Each student is defined by several attributes (e.g. Name, SIN and St\_Num).



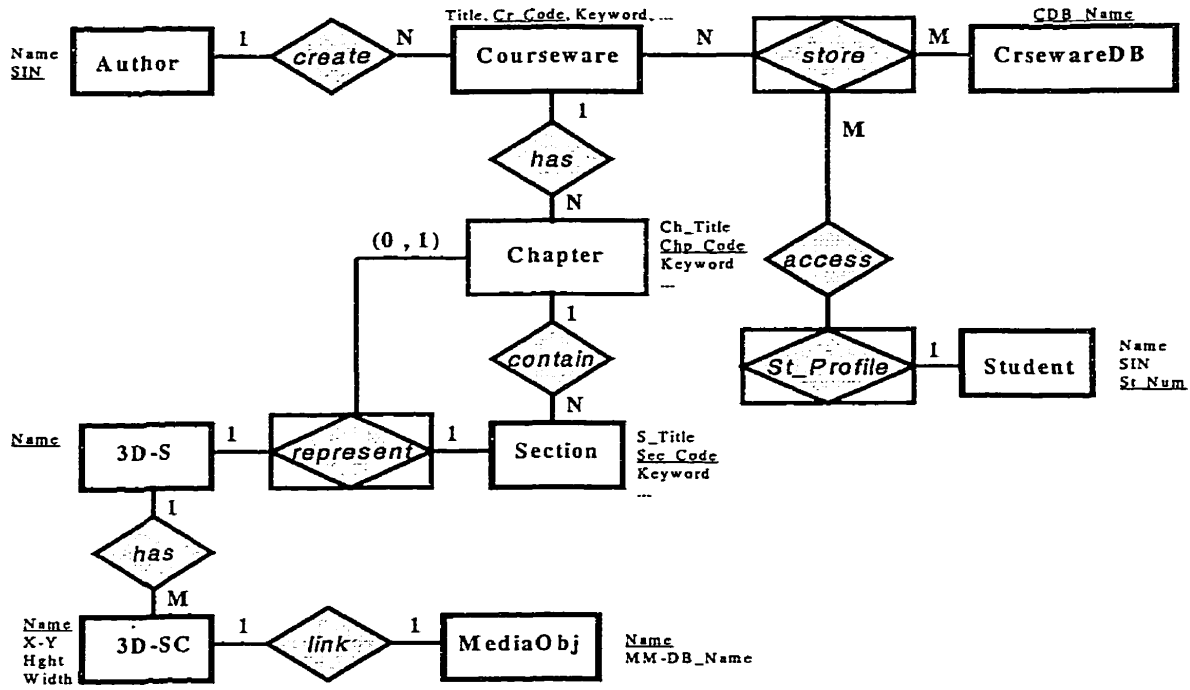


Figure 4.3 MITS E-R Diagram

### 4.3.2 Object-Oriented Modeling

Because of the shortcomings of the Relational-Model to represent the temporal characteristics and behavioral information of each media object in the E-R diagram, we have exploited Object Modeling Technique (OMT) “Rumbaugh’s notations” [18]. The reason being that the OMT notations are capable in presenting both hierarchical and aggregation relationships, and in enhancing the design stage of database schema, as well as generating MITS’s OMT classes. Therefore, we have translated the preliminary E-R diagram (Figure 4.3) to MITS’s OMT diagram (Figure 4.4), and described MITS’s meta data, based on the OMT model. The OMT diagram shows various classes plus their associations, where each box in figure 4.4 represents a class associated with its attributes and methods. It contains various classes such as Doc\_Base, Courseware, Chapter, Section, 3D-S, 3D-SC, StudentActor, CoursewareDB, MM-DB, MediaObj and GUI. It

shows the relative association between classes. For example, class StudentActor interacts with class GUI and class Courseware contains instances of class Chapter. It also represents the class cardinality, as is the case when a Courseware contains one or more chapters, a Chapter has zero or more sections, a Chapter is linked to zero or one 3D-S only, and one 3D-SC is associated with only one MediaObj.

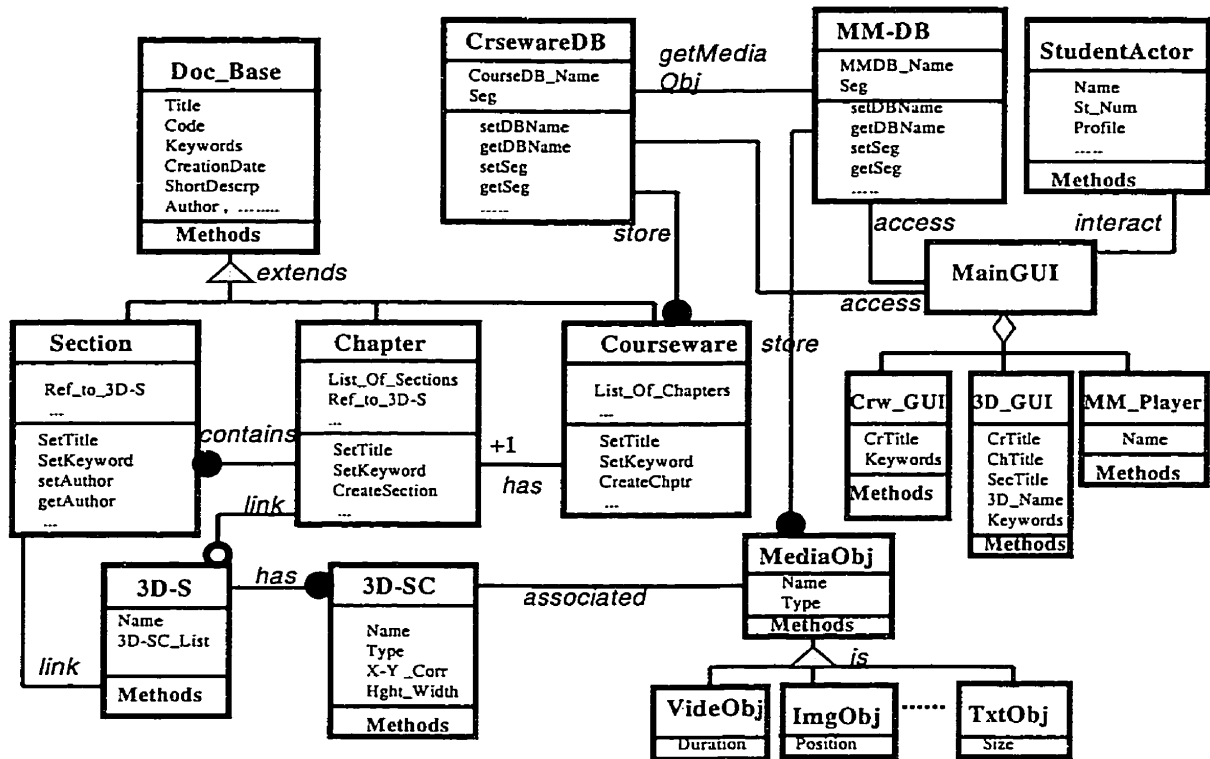


Figure 4.4 MITS OMT Diagram

In addition, MITS's OMT diagram illustrates how MITS exploited the advantages of reuse and inheritance of object-oriented model and OMT notations to share common attributes (title, code, keyword, creation\_date, short\_description, ...) (Figure 4.4), which are defined at Doc\_Base class and inherited in its successor classes (Courseware, Chapter and Section). The OMT model considers Courseware, Chapter and Section classes are extensions of Doc\_Base class and have additional attributes such as List\_of\_Sections and

Ref\_to\_3D-S. The MediaObj class is a generalization of its descendant classes (VideObj, ImgObj, TxtObj,...), which are not overlapped. The MainGUI class is an aggregation of classes (Crw\_GUI, 3D\_GUI and MM\_Player). It identifies the “whole-part” relationship, where the MainGUI class represents the “whole” side, while Crw\_GUI, 3D\_GUI and MM\_Player classes represent the “part” side of the relationship. StudentActor is a class, which simulates an end-user and is defined by St\_Name, St\_Num and is associated with St\_Profile.

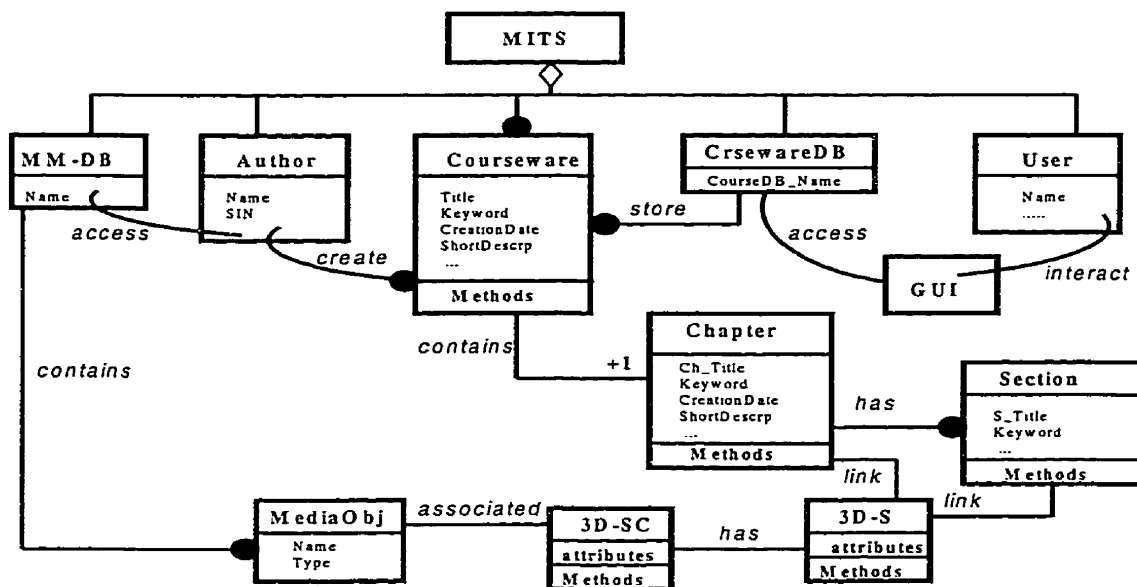


Figure 4.5 Primary MITS OMT Diagram

It is not required that each OMT class at the analysis stage (Figure 4.5) must exist at the design stage (Figure 4.4), or each OMT class must be implemented as a separate class. For instance, class Author at the analysis stage (Figure 4.5) became a multi-valued attribute at the design stage, where such attribute is included within Courseware, Chapter and Section classes that are inherited from Doc\_Base class (Figure 4.4). This represents how the Object-Oriented model could offer reduction in the implementation and how it enhanced the designing stage.

Moreover, we have exploited Jacobson's Use Cases notations [38] to produce MITS Use Cases. This aims to simplify the creation of MITS classes and to clarify their definition (e.g. attributes, role of each method and message passing). Use Cases enable us to understand how classes could communicate and exchange messages. Use Cases are correlated with their *actors*, where each use case has a name and steps. A use case can be defined as a typical sequence of events that can occur when an actor is interacting with the system being modeled. An actor can be defined as a human or machine that will interact with the system being modeled [38]. We have constructed the following use cases:

#### **Use Case 1: Capturing and Preparing Multimedia Information**

Actors: Media Production Center, Media Specialist, Information Sources, Servers

Steps:

- 1- Media Production Center captures real-world information
- 2- Media Production Center analyzes data type (e.g. video, audio, image and text)
- 3- Media Specialist uses MPEG compression format for video data
- 4- Media Specialist uses .RA, .WAVE and .AU compression format for audio data
- 5- Media Specialist uses JPEG compression format for image data
- 6- Media Production Center codes captured information into appropriate media objects
- 7- Media Specialist assigns each media object with a reference name
- 8- **Saving Media Objects in MM-DB Server** (use case)

- **Use Case 2: Saving Media Objects in MM-DB Server**

Actors: Media Production Center, Database Administrator, MM-DB Server, Media Managers

Steps:

- 1- DB Administrator selects a specific MM-DB server or segment
- 2- DB Administrator selects a suitable Media Manager
- 3- DB Administrator stores media objects into MM-DB server
- 4- **System error in MM-DB server** (use case)
- 5- DB Administrator releases resources (i.e. close MM-DB server)

- **Use Case 3: Creating Courseware Data Model**

Actors: Author, Courseware\_DB Server, MM-DB Server, Rendering Tools, Browsers

Steps:

- 1- Creating courseware logical structure
- 2- Linking each section in the logical structure with 3D-S
- 3- Specifying the presentation information in 3D-S
- 4- Each 3D-S is arranged into many 3D-SC, each 3D-SC represents the layout, temporal and behavior structures of a media object
- 5- Each 3D-SC is linked to only one media object
- 6- 3D-S wraps presentation information and interaction behaviors

- **Use Case 4: Creating Courseware**

Actors: Author, Courseware\_DB Server, MM-DB Server

Steps:

- 1- Author analyzes characteristics of courseware users, courseware content and teaching architecture
- 2- Author specifies courseware logical structure
- 3- Author accesses MM\_DB server and selects appropriate media objects
- 4- Author specifies the presentation information of logical structure leaf level
- 5- Author integrates meta data and the selected media objects into a courseware
- 6- Author specifies the state and behavior of a courseware
- 7- **Saving courseware in Courseware\_DB server** (Use Case)

- Use Case 5: **Saving courseware in Courseware\_DB server**

Actors: Database Administrator, Courseware\_DB Server, DBMS

Steps:

- 1- Courseware meta data declared as persistent-capable classes
- 2- Database Administrator selects a specific Courseware\_DB server
- 3- Courseware\_DB segment is selected
- 4- Courseware\_DB segment's root is fetched
- 5- DB Administrator uses update transaction that holds courseware meta data
- 6- Courseware meta data stored in a segment
- 7- Transaction commit
- 8- ObjectStore DBMS releases the update locks related to such segment
- 9- Database Administrator closes Courseware\_DB segment

- **Use Case 6: System Error in Courseware\_DB Server**

Actors: Database Administrator, Courseware\_DB Server, DBMS

Steps:

- 1- Courseware meta data declared as Persistent Capable Classes
- 2- Database Administrator selects a specific Courseware\_DB server
- 3- Courseware\_DB segment is selected
- 4- Courseware\_DB segment's root is fetched
- 5- DB Administrator generates update transaction that holds courseware metadata
- 6- Exception is thrown
- 7- Transaction abort
- 8- Courseware\_DB segment is rolled back

- **Use Case 7: Retrieving a Courseware**

Actors: User, Courseware\_DB Server, MM-DB Server, DBMS

Steps:

- 1- User accesses Courseware\_DB server to browse courses hierarchy
- 2- User selects a specific courseware
- 3- An appropriate transaction is used
- 4- Courseware logical structure cached to the user
- 5- Specific section is selected
- 6- Related presentation information of such section accessed
- 7- Presentation information cached to the user
- 8- Transaction commit

## 9- DB Administrator closes Courseware\_DB segment

Furthermore, for each concrete use case we can draw an *Interaction diagram*. The Interaction diagrams describe how each use case is offered by communicating objects (i.e. use case actors). The diagram shows how the participating objects realize the use case through their interaction [38]. The advantage of using an Interaction diagram is that it is easier to read messages passing in relative order. Thus, Interaction diagrams could be used to reveal how MITS use cases' actors are communicating and exchanging messages in order to improve the designing stage and to produce the actual implemented classes. Finally, we have translated MITS OMT diagram, Use Cases and Interaction diagrams to actual Java classes, which represent MITS database schema. Once the schema is identified, the developer or database administrator can use the DBMS to create a database and populate it. Therefore, OMT classes and Use Cases have been helpful in accomplishing the design stage and generating database schema of MITS system.

## **4.4 MITS Database Management Architecture**

Before describing MITS database management architecture, let's first step back and introduce a brief overview for client-server architecture. At sub-section 4.4.1, we present a brief overview for client-server technology. Then, we describe MITS database management architecture because it is based on this technology.

### **4.4.1 Client-Server Background**

Client-server computing is the technology that helps developers and users to achieve several objectives, as follows: to allow organizations to use networks connecting



different kinds of machines, to lower computing costs by running more of the business on low-cost platforms, and to increase the productivity through user-preferred graphics, interfaces and tools [37].

Client-server computing has both hardware and software implications. The hardware implications are that the computing environment includes desktop machines, networking, and multiple servers, which can be a general-purpose machine or dedicated to specific tasks, e.g. database server, e-mail server or video server etc. While, the software implications are that the contained programming components that can be distributed across machines, e.g. a client software component that invokes the services of one or more server software components [37].

Generally, there is a clear functional separation between a client and a server. Each has a specific functional role, but they interact seamlessly from an application perspective. The application does not have to deal with the processes separately. Client functionality focuses on user interaction; while server functionality makes a system resource available to many clients. A server is able to support multiple clients concurrently. This characteristic implies that servers are shared resources that can be leveraged across applications and users [37].

The different commercial OODBMS products on the market today were architected with a client-server model to provide data to users, applications and tools in distributed computing environments. However, not all OODBMSs implement the same kind of client-server approach. The several approaches have significant variance in their resulting database performance, flexibility, and computing costs.

There are three basic architectural alternatives for implementing client-server functionality in a database manager: the object server approach, the page server approach and the database server approach [37]. They vary in the level of responsibility assigned to the client and server components of the system. In all cases, the client part of the DBMS is linked with the client application process, while the server part of the DBMS is located on the machine where the physical database resides [37].

The OODBMS products available today use either the object server approach or page server approach. Both architectures take advantage of desktop processing power and storage capacities. While, the relational database management system (RDBMS) products use the database server approach, because they were originated in the mainframe and minicomputer eras. Of the three approaches, the object server approach is the most compatible with cooperative, object-to-object processing, where objects are distributed over networks and send messages to each other to invoke each other's services [37].

This is a brief overview for the client-server architecture that is used in OODBMS products.

#### **4.4.2 Description of MITS Database Management Architecture**

Telelearning systems impose database management system to handle full database support for their users by providing efficient storage, effective manipulation, fast querying, rendering courseware media objects and support sharing of courseware components by authors (i.e. teachers) and students. MITS database management system uses 3-tier architecture: client application, application server and the ObjectStore server as illustrated in figure 4.6.

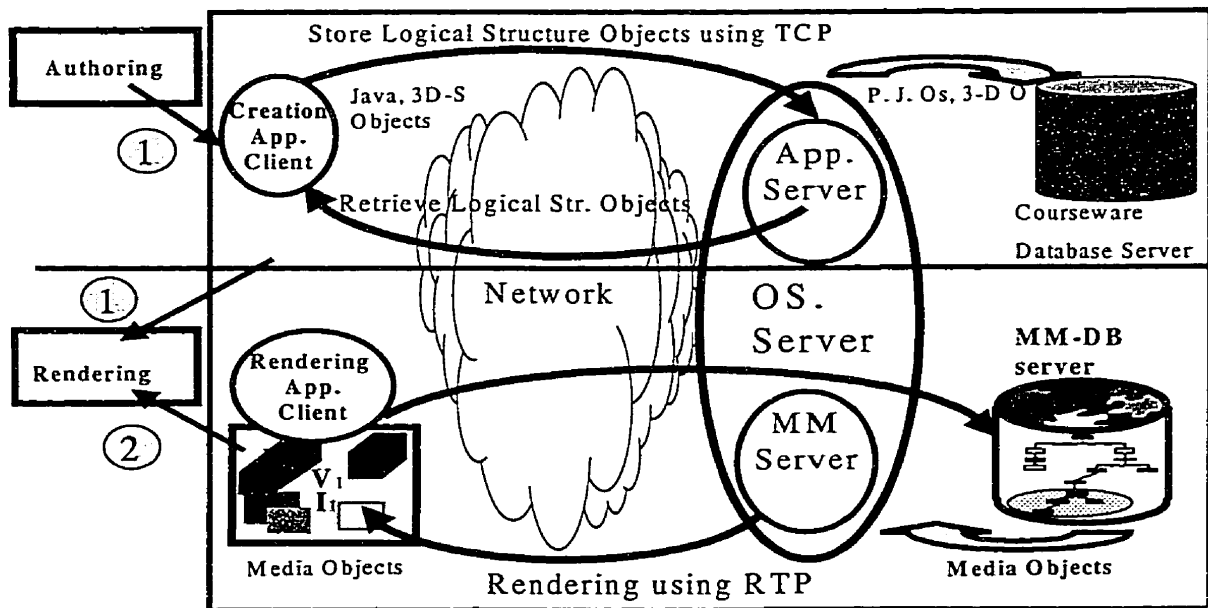


Figure 4.6 MITS Database Management Architecture

It shows two parts of database management, the upper one describes the courseware creation database management, while the lower one describes the courseware rendering database management.

- Courseware creation:** The authoring database management contains client application and application server as two separate entities that are developed, based on Java environment and ObjectStore 5.0. MITS authoring database management is composed of several steps. First, the client application is originated from MITS's database schema as explained in section 4.3. It is called Creation Application Client (CAC), which includes the logical structure classes and 3D-Scenario Structures as volatile Java classes and VRML objects that are transferred to the Application Server (AppSr) for storage using reliable TCP. Second, an ObjectStore API (Class File Postprocessor) will annotate and mark those volatile classes as persistent-capable classes, which means the capacity of classes to be stored in a database [15][19].

Third, **AppSr** is responsible for handling schema operations and queries. It instantiates those persistent-capable classes and generates persistent-capable objects based on schema definitions. Then, it stores them as persistent objects in one CoursewareDB segment or multiple segments, which are managed by the ObjectStore server. Fourth, each persistent 3D-Scenario Structure contains many references to basic media objects. Fifth, these media objects and their references are stored into a MM-DB server using a customized application server called MM Server, which is a component of the lower part of the database management. It works on top of ObjectStore server and exploits the built-in ObjectStore Media Managers (Figure 4.6). In addition, an author can retrieve a courseware for further update or reuse of its components to construct a new courseware, then stores it back at the CoursewareDB server using “update transactions”. Furthermore, database administrators or authors are able to retrieve the database schema for further update or to construct a new schema. Then, annotate all classes using the ObjectStore API “Class File Postprocessor” and utilize the existing database or create a new one, more details of the annotating process will be given at chapter 5.

- **Courseware rendering:** The lower part of figure 4.6 represents the rendering database management. A Rendering Application Client (RAC) is developed using Java environment. It is independent from Creation Application Client (CAC) that satisfied the database schema creation. RAC will ensure courseware delivery to students by communicating with **AppSr** based on ObjectStore server APIs, Media Managers and RTP. When a student wants to access the CoursewareDB server to retrieve a courseware, RAC informs **AppSr** through a request. Then, **AppSr** retrieves

the courseware meta data (logical structure classes and presentation information) and stores them into Client Cache Manager in order to make courseware meta data accessible by a student. The Cache Manager is responsible to ensure concurrent access to data by handling callback messages from the server to client applications [15]. Media Managers will retrieve the associated media objects and start rendering them according to the spatial, temporal and behavioral structures predefined in the 3D-Scenarios. RAC is able to access the **AppSr** for processing queries. RAC sends a request through method-involutions to the **AppSr**. Method-involution will exchange messages between the client and the server by using Java APIs and sockets. A message will convey a student's request for a specific courseware. Then, it is the turn of **AppSr** to interpret the request, locate the database segment, fetch the required courseware, retrieve its objects and locate the associated media objects. Finally, **AppSr** will send over all related courseware meta data and presentation information to RAC using TCP and RTP. The system must ensure the concurrent access of multiple readers to the same courseware.

All implementation issues related to MITS database schema and application-server modules will be described in chapter 5.

## 4.5 Object Persistency States

So far, we have examined MITS database management issues; including the utilized OODBMS, the database modeling as well as schema generation using object-oriented technology, the MITS database management architecture. Finally, we conclude this chapter by presenting the persistency states of objects from ObjectStore point of

view. This clarifies how objects are manipulated inside ObjectStore database. As a basic rule in order to store objects in ObjectStore database, these objects must be persistent-capable. This means that they already have been annotated and contain the required annotation code as explained in sub-section 4.2.3.

Once objects are stored in the database, they are called persistent objects. A persistent object always exists in one of three states: hollow persistent object, active persistent object and stale persistent object as illustrated in figure 4.7 [15].

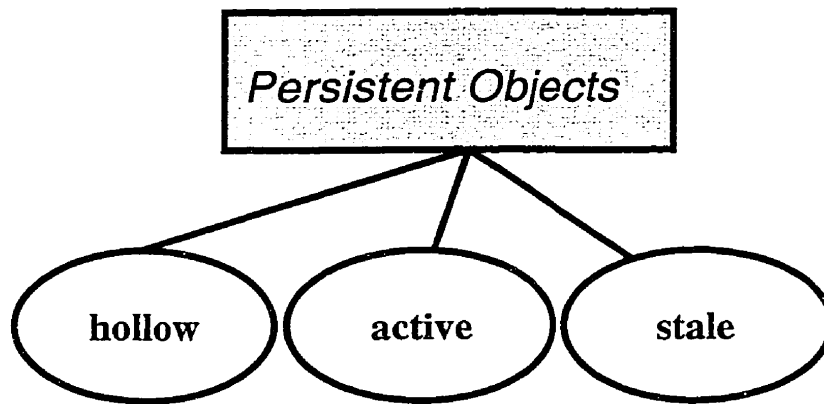


Figure 4.7 Object Persistency States

#### **4.5.1 Hollow persistent object**

A hollow persistent object contains fields that are identical to the fields of the object in the database that the persistent object represents, but the fields have default values. When an application acquires a reference to an object that has not yet been read in from the database, ObjectStore generates a hollow object as a placeholder for that object. ObjectStore does not actually read in the contents of the object until the application tries to access the object. When the application reads or updates a hollow object, ObjectStore turns it into an active persistent object [15].

## **4.5.2 Active persistent object**

An active persistent object starts as an exact copy of the object that it represents in the database. The contents of an active object are available to be read by the application and might be available to be modified. If an active object is updated by the application, it is no longer identical to the object in the database that it represents. An application can read or update an active persistent object, a persistent object must be active for an application to read or update it by utilizing appropriate transactions [15].

## **4.5.3 Stale persistent object**

A stale persistent object is no longer valid. Its fields have default values and should not be used. A persistent object becomes stale after the application calls specific APIs; for instance: `ObjectStore.destroy()`, `Transaction.commit()`, `Transaction.abort()`. There are several rules for utilizing such APIs. If an application tries to read or update a stale object, an appropriate exception will be thrown. There is an important rule for using `ObjectStore` “any application must not invoke any `ObjectStore` operation on a stale object” [15]. Therefore, main operations such as update and retrieve are performed on active objects.

# Chapter 5

## 5 System Implementation

### 5.1 Introduction

In this chapter, we will tackle the implementation issues of MITS database system. First, a brief overview of the utilized Object-Oriented programming environment, “Java environment”, as well as its important features will be addressed. Then, MITS database system implementation is described. This implementation is divided into two major tasks: the implementation of database schema and the implementation of application-server engine as shown in figure 5.1. The implementation of application-server is the main contribution of the chapter, and it covers all concepts that have been described so far. The application-server has been developed as a set of modules. It imposes the utilization of database schema implementation, thus, the database schema implementation will be described prior to the description of the application-server implementation.

A proposed template for implementing additional modules will be described in section 5.4. In addition, the generated schema classes and application-server modules must be annotated using an ObjectStore API. The annotation process for MITS classes will be described in section 5.5. Section 5.6 presents samples of the results and shows how these modules are utilized to construct, retrieve as well as update the meta data of several courses. Finally, as we have pointed out that MITS consists of several major



components, we will address the integration issue of the database engine with other MITS system's components such as rendering application and system's GUIs.

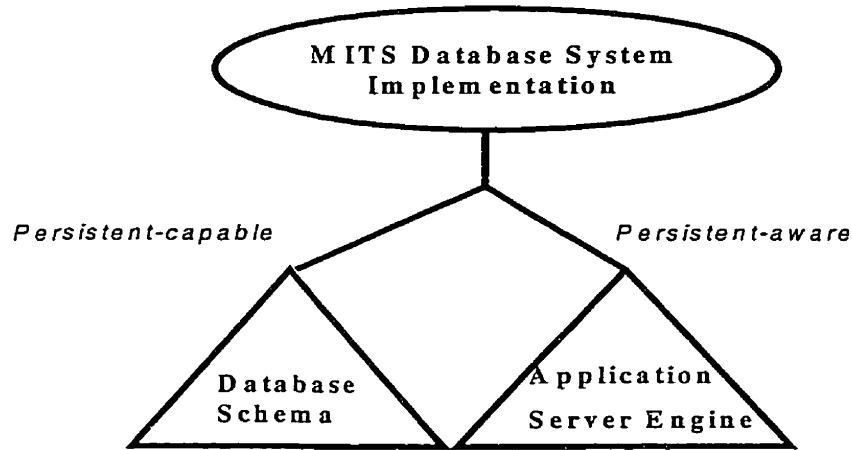


Figure 5.1 MITS Database System Implementation

Therefore, in order to fulfill the implementation tasks, we have exploited the state-of-the-art technologies by utilizing Java JDK 1.1.2 [39] and ObjectStore 5.0 in Java interface 1.05 [40] that operate on Windows NT 4.0 [22].

Although, the database schema classes and application-server modules can be implemented in C++ in order to provide a high performance, Java was used for the prototype to make system modules platform-independent, accessible through the Internet by a number of educational groups and executable on Web browsers across the network.

## 5.2 Java Environment

Java is an Object-Oriented programming language that is created by Sun Microsystems [39]. Sun itself describes Java as follows: “Java is a simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multithreaded and dynamic language” [41]. Java is simple in the sense that

it is easy and quick to learn compared to other programming languages. It resembles C++ but omits many confusing features of C++ that bring more grief than benefit, such as pointers and operator overloading [41]. In addition, Java was designed to be object-oriented from the beginning. It emphasizes on a basic object-oriented principle “*PIE principle*”, which stands for polymorphism, inheritance and encapsulation [42]. These concepts are fundamental ones for Object-Oriented programming environments. One of the benefits of the PIE principle is to sustain code reuse by allowing software components to grow from existing components, which reduces the required time to create new components. Moreover, one of the aims is to make Java support distribution. It is designed to run applications on networks. Thus, it provides different libraries and classes for network connectivity, such as sockets [41]. Since, the network environment has been kept in mind when standard Java libraries have been developed, so programming client-server applications with Java is easier than with any other programming language [43]. Furthermore, Java compiler generates byte-codes instead of native machine code [42]. A Java interpreter is exploited to run the byte-code. A Java program can run on any platform that has a Java interpreter and run-time system, known together as “Java Virtual Machine” [44], which makes Java platform-independent. What makes Java perfect for the Internet programming is the relatively small size of the compiled byte-codes.

Another important feature of Java is “*multithreading*”. It is a way of building applications with multiple processes or threads [41]. Unfortunately, writing programs that deal with many things happening at the same time can be more difficult than writing in the traditional single-threaded C and C++ style. Java has a sophisticated set of synchronization primitives that support multithreading. Other advantages of

multithreading are better interactive responsiveness and real-time behavior. However, this is restricted by the underlying platform [41].

Java contains different packages that could facilitate programming tasks such as I/O, networking, graphical-user-interface components, etc. Other advanced tools, such as Java-enabled-browsers, applets and Servlets are causing Java to become a standard for Internet programming [41][44]. These tools have been utilized in our system's development and integration as will be addressed briefly in section 5.6.

### **5.3 Implementation of MITS Database Schema**

So far, as we have seen in chapter 4, we have exploited Object-Oriented paradigm to model and design MITS database schema (Figure 4.4). OMT notations and Use Cases have been used to design the schema. In this section, we describe the translation of the database schema to actual Java classes and address the schema evolution in sub-sections 5.3.1 and 5.3.2 respectively.

#### **5.3.1 MITS Schema Classes**

The implementation of database schema consists of several Java classes: *Doc\_Base*, *Courseware*, *Chapter*, *Section*, *Scenario*, *SComponent*, *Head*, *Body* and *Tail*. Each one of them contains several public as well as private attributes. In addition, multiple public methods are included within each class. We have implemented schema classes by utilizing regular Java libraries plus importing the necessary ObjectStore APIs, which are essential for the annotation process, as will be explained in section 5.5.

- Class `Doc_Base`: is an abstract class that contains several attributes and methods inherited in its successor classes (`Courseware`, `Chapter` and `Section`). These attributes include `title`, `code`, `author`, `creationDate`, `modDate`, `keywords[]`, and `shortDescription`. It also includes several `set_methods` (e.g. `setTitle`, `setCode`, `setAuthor`, `insertKeywords`, `deleteKeywords`, etc) and `get_methods` (e.g. `getTitle`, `getCode`, `getAuthor`, `getKeywords`, `getDescription`, etc). This class illustrates how Object-Oriented languages support the reuse of the code and offer reduction in the implementation (reduce the cost of implementation).
- Class `Courseware`: is an extension of `Doc_Base` class. It is intended to create persistent instances of such class, which are stored in the `Courseware Database` server. Class `Courseware` includes the above mentioned attributes and methods that are inherited from the super class `Doc_Base`. It also includes `list_of_Chapters` as an attribute. In addition, it contains multiple methods such as `insertNewChapter`, `setChapter`, `deleteChapter`, `getChapter`, `getChapters` and `getChapterPosition`.
- Class `Chapter`: is an extension of `Doc_Base` class. It is intended to create persistent instances of such class that are stored within each `courseware` instance, then store `courseware` instances in the `Courseware Database` server. Class `Chapter` includes the above mentioned attributes as well as methods and also includes the following attributes: `status`, `list_of_Sections` and `a3D_S`. If *chapter's status* equals `false`, this means that a `Chapter` instance has no section and it has a link to an instance of `Scenario` class, which is `a3D_S`. In addition,

class Chapter contains several methods, such as insertNewSection, setSection, deleteSection, setStatus, insert3D\_S, delete3D\_S, getSection, getSections, getStatus, get3D\_S and getSectionPosition.

- Class Section: is an extension of Doc\_Base class. It is intended to create persistent instances of such class that are stored within each persistent chapter instance, which are contained at a courseware instance in the Courseware Database server. Class Section includes the above mentioned attributes and methods. It also includes the following attributes: refList and a3D\_S. In addition, it contains several methods such as insertMediaRef, setMediaRef, insert3D\_S, delete3D\_S, getMediaRef and getRefList.
- Class Scenario: represents a 3D\_S that wraps the presentation information of different media objects as we have designed in our Courseware Data Model. It is intended to create a persistent instance of such class linked with a section/chapter instance in order to structure the presentation information of its media objects. It contains private attributes such as name and scmpList. The attribute scmpList is a list of 3D\_SComponents that are wrapped in a Scenario instance. In addition, it includes multiple methods such as setName, getName, insertScmpList, deleteScmpList and getScmpList.
- Class SComponent: represents a 3D\_SComponent, where each instance of this class models a physical media object. It includes different attributes such as name, layout information of each media object (e.g. x-y values, height-width measurements), head, body, tail and mediaName. In addition, it contains

several methods such as setName, setX\_Value, setY\_Value, setHeight, setWidth, setMediaName, getName, getX\_Value, getY\_Value, getHeight, getWidth, getMediaName, getHead, getBody and getTail.

- Classes Head, Body and Tail: represent the temporal-ordering plus the synchronized list of media objects with a specific media object. Class Head contains the startTime and syncList as private attributes. The startTime attribute represents the starting time for rendering a specific media object. While, the syncList attribute represents a list of references for all media objects that are synchronized with the current media object. In addition, it contains some methods to handle these attributes such as setStartTime, insertSyncList, deleteSyncList, getStartTime and getSyncList. Moreover, Class Body contains the duration and syncList as private attributes. The duration attribute specifies the actual time for rendering a media object and how synchronized its playback with other media objects. Furthermore, Class Tail contains endTime and syncList attributes. The endTime attribute represents the ending time for rendering a specific media object, while, the syncList attribute identifies all media objects that are synchronized with the current media object. These media objects have the same endTime, which means that they stop rendering at a specific time. Appropriate methods are implemented in both classes Body and Tail to handle these attributes.

### 5.3.2 Schema Evolution

In general, a database schema is specified during database design and is not expected to change frequently, but the actual data in a database may change relatively frequently [14]. In other words, the schema does not change so often because it describes the structure of persistent-capable classes. While, the persistent objects may change by updating their meta data values and/or adding or removing persistent objects to the database.

As the number of applications that access a given class of objects grows, there is usually a need to modify the structure of the objects to better meet the needs of the applications. If the structure of a class of persistent objects is changed, all instances of such class are affected. Such process of changing the structure or the behavior of persistent classes is called *Schema Evolution* [36].

Therefore, ObjectStore is one of those OODBMSs that support schema evolution. It provides a number of APIs for migrating old objects from old databases to conform to the newer schema, and it also provides some facilities by which applications compiled with the newer class definitions could access existing databases [15][36].

## 5.4 Implementation of Application-Server

### 5.4.1 Application-Server Modules

The main contribution of this chapter is the implementation of the **application-server** (Figure 5.1). It aims at creating, populating, accessing and updating the database.

It also supports retrieving and deleting persistent objects from the database. In order to satisfy such tasks, we have classified them into several modules. Each one is responsible for performing a particular task. Since, we have designed and implemented our system based on Object-Oriented paradigm, so we have implemented the application-server as a set of modules, where each independent module is dedicated for achieving a specific task. These modules include the CourseDB module, updateDB module, retrieveDB module and deleteDB module (Figure 5.2). These modules perform their tasks on a collection of courses. In our case, we have used ObjectStore\_Vector to contain the collection of persistent courses. Therefore, we have developed a class called *theCollection* as a wrapper class, which provides the required methods to handle the management of persistent courseware instances.

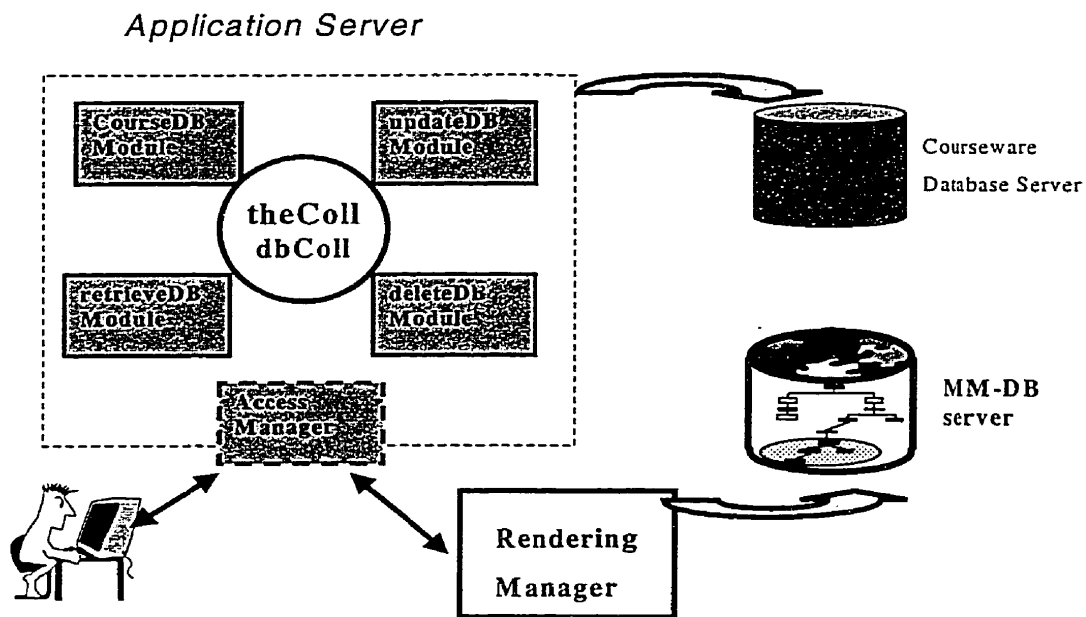


Figure 5.2 Application-Server Engine Modules



- TheCollection class is employed as a data structure to wrap courseware instances and store them permanently (Figure 5.2). It uses ObjectStore\_Vector as the actual data structure. We have developed several methods to make such class as the container of courseware objects based on Java environment and ObjectStore APIs. These methods include:

*- public Courseware getCourse (String aCode);*

*- public void addCourse (Courseware cr);*

*- public Enumeration getAllCourses ();*

*- public void removeCourse (String aCode);*

- The CourseDB module is a part of the application-server. This module is responsible for initializing, creating and populating the database. It uses ObjectStore APIs to initialize ObjectStore server and start a database session. Then, it creates the database according to a particular mode of reading as well as writing to the database. If any system error happens or something goes wrong, an appropriate exception handler will be employed, and the database will be closed. Otherwise, a suitable transaction “update transaction” is started in order to create courseware instances and populate the database. The constructed courses are instances of one of the database schema classes, i.e. “Courseware class”. These instances are stored in the wrapper class “theCollection” (Figure 5.2). It is up to the database administrator to decide how many courses to be created. Once he/she accomplishes the creation

process, ObjectStore API “*Database.createRoot*” is used to create a *Root*, which is a reference that points to the *Collection* inside the constructed database. This root is essential to access the persistent objects by using other modules. Finally, the transaction is committed and those courseware objects are permanently stored. This brings the database into a consistent state. In this module, we have implemented several methods to facilitate the creation of the database and satisfy the instantiation of courseware objects. These methods include *newCourse()*, *acceptData()*, etc.

- The updateDB module is the largest module of the application-server. It is responsible for updating the database by modifying the meta data of persistent courseware objects. In addition, it is responsible for inserting new objects such as courseware, chapter and section instances into the database. This module follows the same sequence of events that the CourseDB module has followed in order to accomplish the task. These events include initializing ObjectStore server and starting the session, and then opening the database according to a specific mode of “*Database.open()*”. A proper transaction is started to access the database root and navigate from such root to locate persistent objects. If any system error happens an appropriate exception will be thrown and the database will be rolled back in order to be retained in a consistent state. As pointed out in chapter 4, at the beginning, ObjectStore server sends hollow versions of the persistent objects. Once an object is under an update action, ObjectStore will provide an active object to be updated. Thus, the updateDB module works within a transaction on active objects.

Finally, either the transaction is committed and those updates will be reflected in the database, or the transaction is aborted and the database will be rolled back in order to remain in a consistent state. In this module, we have developed several methods to handle updating and accessing operations on the persistent objects within a transaction. These methods include:

- *public static void options (theCollection dbColl);*
- *public static void workOnCourse (theCollection dbColl);*
- *public static void addCourseColl (theCollection dbColl, Courseware cr);*
- *public static void updateCourseware (Courseware cr);*
- *private static void acceptData ();*

In addition, other methods are also implemented to accomplish the complex task of the updateDB module.

- The retrieveDB module is a part of the application-server, which is responsible for accessing and retrieving persistent objects inside the database by navigating through the database root using a suitable transaction mode. We have implemented several methods to accomplish the retrieving task, e.g. *showData (theCollection dbColl)*. This module works on the persistent collection to browse courses, retrieve a specific courseware and retrieve a particular section. It can be extended to serve other kinds of queries by developing new methods to perform other kinds of retrieve.

- The deleteDB module is a part of the application-server. It is responsible for deleting a particular courseware from the database collection and ensuring the deletion of all of its internal objects to avoid accessing unreachable objects or *stale objects* that are explained in chapter 4. This module follows the same framework of the previous modules. It will initialize ObjectStore, start a session, open the database using an update transaction, fetch the database root, access the required courseware to be deleted, remove all courseware internal objects by utilizing an appropriate method *removeCourse()* from theCollection class, remove the courseware reference from theCollection using ObjectStore API, commit the transaction and then close the database. Once a system error happens a proper exception will be thrown and the database will be brought to a consistent state. We have implemented several methods to manage this task, such as:

- *public static void deleteFromColl (theCollection dbColl, String aCode);*

- *public static void deleteCourseInternal (Courseware cr);*

These modules represent the MITS database system engine, which is transparent to end-users. All these modules are accessed by the database administrator and listen to students' requests through a mediator module called "Access Manager", which is responsible for accepting students' requests, then accessing the database to retrieve the required courses and caching them to end-users' machines. It cooperates with another module "Rendering Manager" to support the delivery of actual media objects to end-users. These two mediator modules are described and developed by [31] at our

laboratory. We will address the integration issue of the database engine with these mediator modules at section 5.6.

### **5.4.2 Template for Implementing Additional Modules**

Since, we have designed and implemented our system based on Object-Oriented paradigm and because of its flexibility to add new components without affecting the previous ones, so it is possible to develop a new module(s) that achieves a specific task, which is not tackled by our existing modules. Therefore, we present a *template* for implementing such additional modules if they are needed. The implementation of any module follows a particular sequence of events (Figure 5.3). First initialize ObjectStore server and start a session, specific exception handlers must be applied to handle system errors or ObjectStore failure. Second, open or create the database depending on the task that is supposed to be accomplished. Third, start an appropriate transaction to perform specific actions utilizing a proper transaction mode. Fourth, fetch the database root. Fifth, implement the main task that is required. Sixth, create the database root if not already created or set. Seventh, commit the transaction and then close the database. Among all those events proper exceptions must be thrown to avoid system errors and to retain the database in a consistent state.

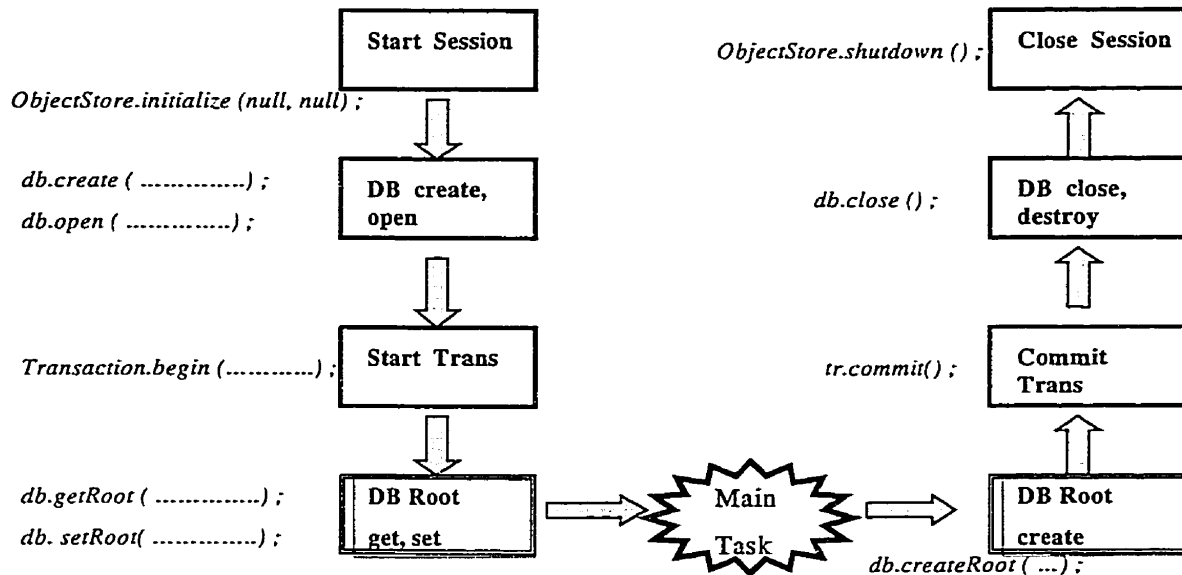


Figure 5.3 Template for implementing application-server modules

## 5.5 Annotating MITS Classes

As we have pointed out in chapter 4, in order to store objects in ObjectStore database, these objects must be persistent-capable. Thus, MITS classes, both schema classes and application-server modules must be postprocessed according to appropriate modes using “Class File Postprocessor” in a process called annotation process. All our schema classes are postprocessed to be persistent-capable including theCollection class, because these classes will be instantiated and stored in the database. The application-server modules are postprocessed to be persistent-aware, because they can manipulate persistent objects of the schema, but themselves are not persistent objects in the database. We have annotated MITS classes using the following Java as well as ObjectStore command:

```
➤ osjcfp -dest ..\osjcfpou\Mits -pc schema.class -pa modules.class
```

This postprocessing command creates two separate directories; one contains the source code classes, while the other one is the destination directory “Mits Directory”, which contains the annotated classes (Figure 5.4).

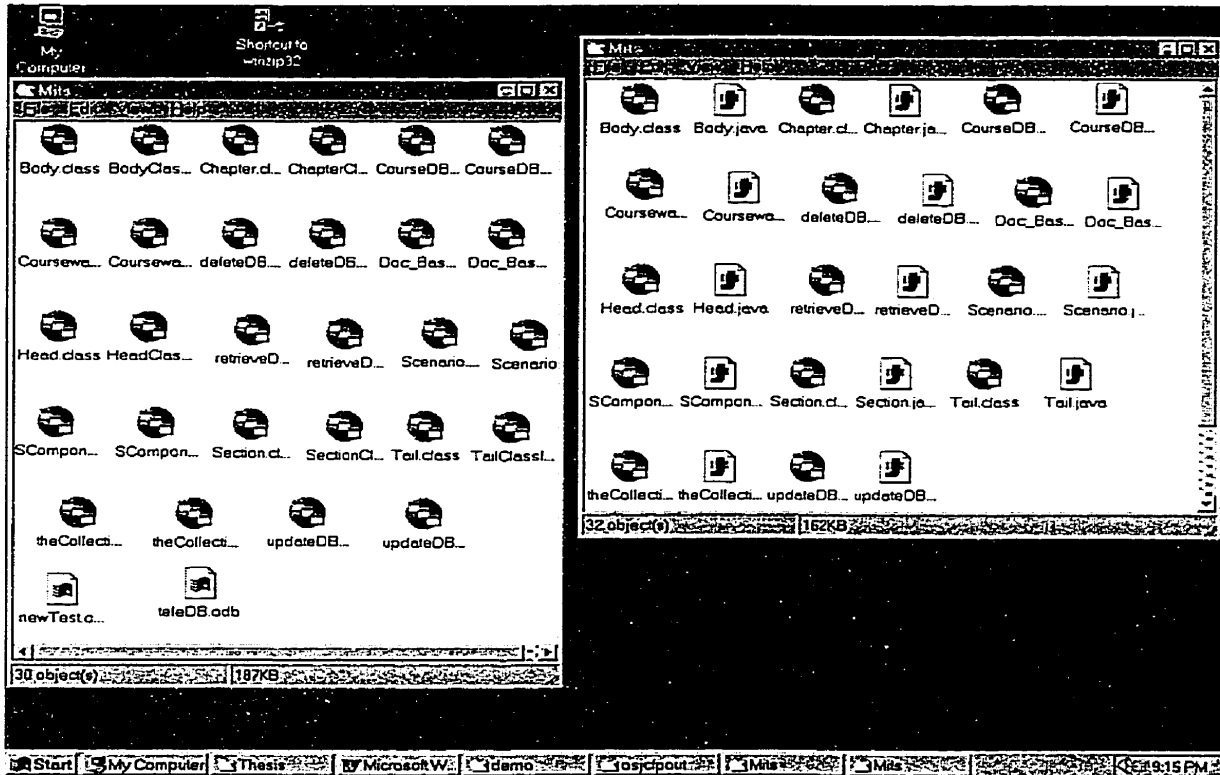


Figure 5.4 Original MITS classes (right), and the annotated classes (left)

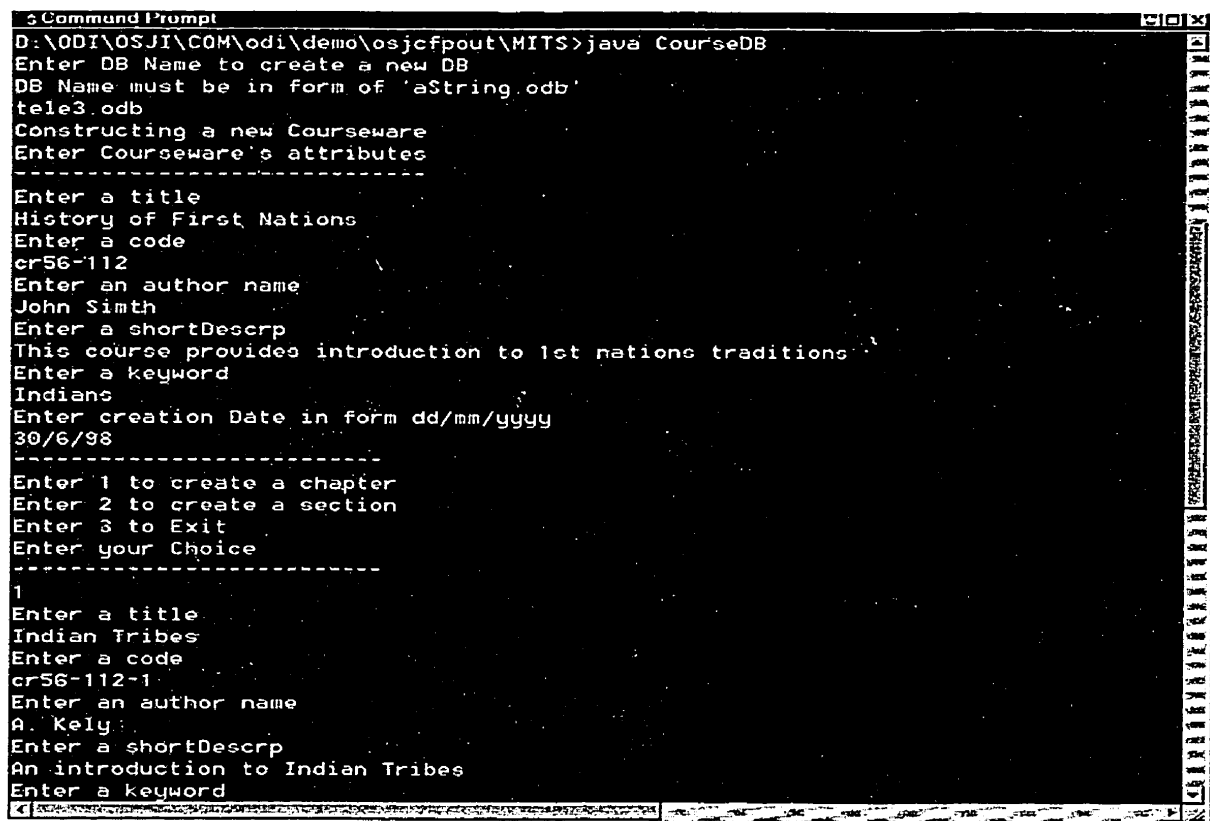
As illustrated in figure 5.4, the annotated classes are shown in the left screen shot. It also contains two databases that are created by using the CourseDB module, and can be updated by utilizing other modules such as the updateDB module.

Prior to this process, class path and system configuration must be set properly to include ObjectStore as well as Java JDK, and utilize the Class File Postprocessor.

## 5.6 Samples of Results

The application-server modules have been used to create, populate and update different databases. For instance, teleDB.odt, tele3.odt and newTest.odt demonstrate the execution of such modules, as shown in figures 5.4 and 5.7.

We have constructed several courses, where their meta data has been stored in teleDB.odt. The teleDB.odt contains various courses, such as “Multimedia Database” and “Logic Programming”. Also, the tele3.odt is created by using the *CourseDB module*. A courseware entitled “History of First Nations” has been constructed as shown in figures 5.5 and 5.6.



```
Command Prompt
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>java CourseDB
Enter DB Name to create a new DB
DB Name must be in form of 'aString.odt'
tele3.odt
Constructing a new Courseware
Enter Courseware's attributes
-----
Enter a title
History of First Nations
Enter a code
cr56-112
Enter an author name
John Smith
Enter a shortDescrip
This course provides introduction to 1st nations traditions
Enter a keyword
Indians
Enter creation Date in form dd/mm/yyyy
30/6/98
-----
Enter 1 to create a chapter
Enter 2 to create a section
Enter 3 to Exit
Enter your Choice
-----
1
Enter a title
Indian Tribes
Enter a code
cr56-112-1
Enter an author name
A. Kely
Enter a shortDescrip
An introduction to Indian Tribes
Enter a keyword
```

Figure 5.5 A Snapshot for Creating a Database and Constructing a Courseware



```
Command Prompt
Enter a keyword
Indians
Enter creation Date in form dd/mm/yyyy
22/7/98
-----
Enter 1 to create a chapter
Enter 2 to create a section
Enter 3 to Exit
Enter your Choice
-----
1
Enter a title
Indian Languages
Enter a code
cr56-112-2
Enter an author name
Z. Sareen
Enter a shortDescrp
The Indian Languages
Enter a keyword
Indian Language
Enter creation Date in form dd/mm/yyyy
11/9/97
-----
Enter 1 to create a chapter
Enter 2 to create a section
Enter 3 to Exit
Enter your Choice
-----
3
To add a new Course to the Collection
Enter Y for yes, To exit Enter N for No
n
D:\ODI\OSJI\COM\odi\demo\oc\jcfpout\MTS>
```

Figure 5.6 A Snapshot for Constructing a Courseware

All the courseware attributes as well as logical structure have been specified and persistently stored. The courseware attributes such as title, code, author, shortDescription, etc, are provided as ObjectStore Java strings. In addition, the courseware hierarchy is defined by specifying its various chapters and sections. Each section contains several textual references to media objects besides the other attributes that define each specific section.

Several databases can be created using *CourseDB module*, as shown in figure 5.7. Each database is populated by multiple courses, and then, committed and brought to consistent state.

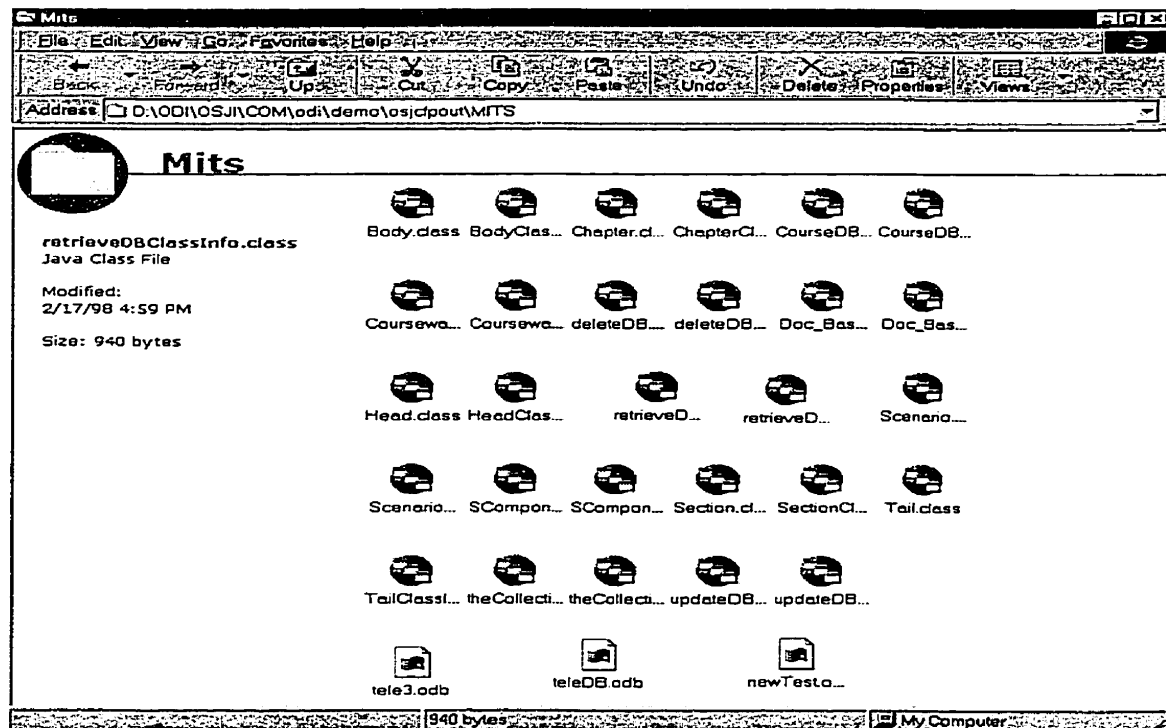


Figure 5.7 Different Generated Databases

As shown in figure 5.8, the execution of the *retrieveDB module* demonstrates how many courses can be retrieved from the teleDB.odt. The logical structure of such courses is cached in order to be delivered to students through the presentation GUIs, which are developed by other member in our laboratory [31].

The *retrieveDB module* delivers the entire logical structure to a bridge-application that responds to students' requests. This improves the database performance by reducing the retrieval time for the related data, since the meta data are read with one instead of several disk reads.

```

Command Prompt
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>java retrieveDB
Course title : cr101 , and has code : cr101
Course Author : ali , and has Discription : this is the 1st courseware
-----
Chapter title : introduction chapter , and has code : cr101-1
Chapter title : Chapter 2 in Course 1 , and has code : cr101-2
Chapter title : chapter 3 DB , and has code : cr101-3
Chapter title : Video Indexing , and has code : cr101-4
Chapter title : DB Modeling , and has code : cr101-5
-----
Course title : Multimedia DB , and has code : cr512
Course Author : Karmouch , and has Discription : This is advanced DB course
-----
Chapter title : introduction to DB , and has code : cr512-1
Chapter title : Multimedia Synchronization , and has code : cr512-2
Chapter title : MM Communication , and has code : cr512-2
-----
Course title : Logic Programming , and has code : cr240
Course Author : Logard , and has Discription : This is a new testing course.
-----
Chapter title : Logic Operations , and has code : cr240-1
-----
Course title : Logic Programming , and has code : cr444
Course Author : Amin , and has Discription : this is prolog programming
-----
Chapter title : intro , and has code : cr444-1
-----
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>

```

Figure 5.8 The Retrieval of Courseware Meta Data

The *updateDB* module is responsible for updating the database by creating a new courseware or modifying an existing one. A new courseware can be created and stored in the current database (i.e. teleDB.odb). In addition, the database administrator could specify the code of a specific courseware that needs to be updated. There are several options for updating an existing courseware, for example adding a new chapter or section, modifying an existing chapter or section, and deleting an existing chapter or section, as shown in figures 5.9a and 5.9b. We have added a new chapter to an existing courseware, which is called “Multimedia Database”. The new added chapter is entitled “Mobile Agent”. The entire attributes of such new chapter are specified and successfully stored in the teleDB.odb, as shown in figure 5.10.

```

C:\Command Prompt - java updateDB
D:\ODI\OSJI\COM\odi\demo\osjcfout\MITS>java updateDB
-----
Enter 1 to create a new Courseware
Enter 2 to update an existing Courseware
Enter 3 to Exit
Enter your Choice
-----
2
Enter Courseware's Code that you want to
update its meta data inside the dbCollection
-----
cr512
Enter Courseware's attributes that you want to update
The following Courseware's attributes are optional
to update. If You Do Not Prefer to update any one
of them, just press 'Enter' to keep the old value!!!
-----
Enter a title

Enter a code

Enter an author name

Enter a shortDescrp

Enter a keyword

Enter a Date in form dd/mm/yyyy

-----
Enter 1 to create a new chapter
Enter 2 to create a new section

```

Figure 5.9a The Updating of Courseware Meta Data

```

C:\Command Prompt - java updateDB
-----
Enter 1 to create a new chapter
Enter 2 to create a new section
Enter 3 to update an existing chapter
Enter 4 to update an existing section
Enter 5 to delete an existing chapter
Enter 6 to delete an existing section
Enter 7 to Exit
Enter your Choice
-----
1
Enter the new Chapter's attributes
Enter a title
Mobile Agent
Enter a code
cr512-4
Enter an author name
Uu
Enter a shortDescrp
This is an introduction to Mobile Agents
Enter a keyword
Mobile Agent
Enter a Date in form dd/mm/yyyy
12/8/98
-----
Enter 1 to create a new chapter
Enter 2 to create a new section
Enter 3 to update an existing chapter
Enter 4 to update an existing section
Enter 5 to delete an existing chapter
Enter 6 to delete an existing section
Enter 7 to Exit
Enter your Choice
-----
7

```

Figure 5.9b The Updating of Courseware Meta Data

The new chapter “Mobile Agent” is stored in the database and can be retrieved as shown figure 5.10.

```
Command Prompt
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>java retrieveDB
Course title : cr101 . and has code : cr101
Course Author : ali . and has Discription : this is the 1st courseware
-----
Chapter title : introduction chapter . and has code : cr101-1
Chapter title : Chapter 2 in Course 1 . and has code : cr101-2
Chapter title : chapter 3 DB . and has code : cr101-3
Chapter title : Uideo Indexing . and has code : cr101-4
Chapter title : DB Modeling . and has code : cr101-5
-----
Course title : Multimedia DB . and has code : cr512
Course Author : Karmouch . and has Discription : This is advanced DB course
-----
Chapter title : introduction to DB . and has code : cr512-1
Chapter title : Multimedia Synchronization . and has code : cr512-2
Chapter title : MM Communication . and has code : cr512-2
Chapter title : Mobile Agent . and has code : cr512-4
-----
Course title : Logic Programming . and has code : cr240
Course Author : Logard . and has Discription : This is a new testing course. it is
-----
Chapter title : Logic Operations . and has code : cr240-1
-----
Course title : Logic Programming . and has code : cr444
Course Author : Amin . and has Discription : this is prolog programming
-----
Chapter title : intro . and has code : cr444-1
-----
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>
```

Figure 5.10 The Retrieval of Courseware After Updating the Database

The *deleteDB module* is responsible for deleting an entire courseware. The database administrator specifies the code of the courseware that needs to be deleted. Since each courseware contains several persistent objects, so these objects must be deleted prior to the deletion of the courseware attributes. In this module, we have provided the necessary methods that ensure the deletion of courseware objects (i.e. chapters and sections) before the deletion of the courseware, in order to avoid accessing *Stale Objects*. We have decided to delete a courseware that had code “cr444”, as shown in figure 5.11, such courseware had been deleted.

```

Command Prompt
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>java deleteDB
Enter Courseware's Code that you want to delete
-----
cr444
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>java retrieveDB
Course title : cr101 , and has code : cr101
Course Author : ali , and has Discription : this is the 1st courseware
-----
Chapter title : introduction chapter , and has code : cr101-1
Chapter title : Chapter 2 in Course 1 , and has code : cr101-2
Chapter title : chapter 3 DB , and has code : cr101-3
Chapter title : Uideo Indexing , and has code : cr101-4
Chapter title : DB Modeling , and has code : cr101-5
-----
Course title : Multimedia DB , and has code : cr512
Course Author : Karmouch , and has Discription : This is advanced DB course
-----
Chapter title : introduction to DB , and has code : cr512-1
Chapter title : Multimedia Synchronization , and has code : cr512-2
Chapter title : MM Communication , and has code : cr512-2
Chapter title : Mobile Agent , and has code : cr512-4
-----
Course title : Logic Programming , and has code : cr240
Course Author : Logard , and has Discription : THIS is a new testing course. it is
-----
Chapter title : Logic Operations , and has code : cr240-1
-----
D:\ODI\OSJI\COM\odi\demo\osjcfpout\MITS>

```

Figure 5.11 The Deletion of an Entire Courseware

The previous snapshots present samples of the results that are stored in different databases. These courses are delivered to students by using graphical presentation user-interfaces, which are developed by [31]. In the next section we will address the integration of the database engine modules with the presentation GUIs.

## 5.7 System Integration

As we have indicated in chapter 2, the goal of the system is to build a seamless education environment that supports the delivery of courses to distributed users over the network. The challenge is to develop a successful Telelearning system that should be

available to multiple distributed educational groups on the network, regardless of the utilized platforms. Thus, the increased popularity of the Internet and its usage as a communication technology have offered an appropriate chance to develop such Telelearning system to help students to access courses from any access point at any time.

As mentioned earlier, our system consists of several implemented modules and components such as database schema classes, application-server modules, rendering application manager, GUIs, authoring components, etc. Therefore, these components must be integrated to achieve the goal of the system. Since, we have exploited the latest technologies and open programming environment “Java platform”, so we have integrated these components utilizing advanced features and tools such as Java-enabled browsers, applets and Servlets (Figure 5.12).

We have utilized Java environment to implement database schema and application-server modules. In addition, a sub-system of MITS called Multimedia Interactive Courseware Rendering System (MICRS) is developed using Java applets and Servlets [31]. It is responsible for rendering multimedia courseware material over the Internet and offers students the required GUIs to access courseware database without the need for installing any additional software packages on their machines. Students need to have only standard browsers to access MITS using the rendering application over the Internet. The only requirements are Java-compatible interpreter and network connections.

This supports the distribution feature of the system. However, a completely distributed system is not possible, because of the networking restrictions and software compatibility.

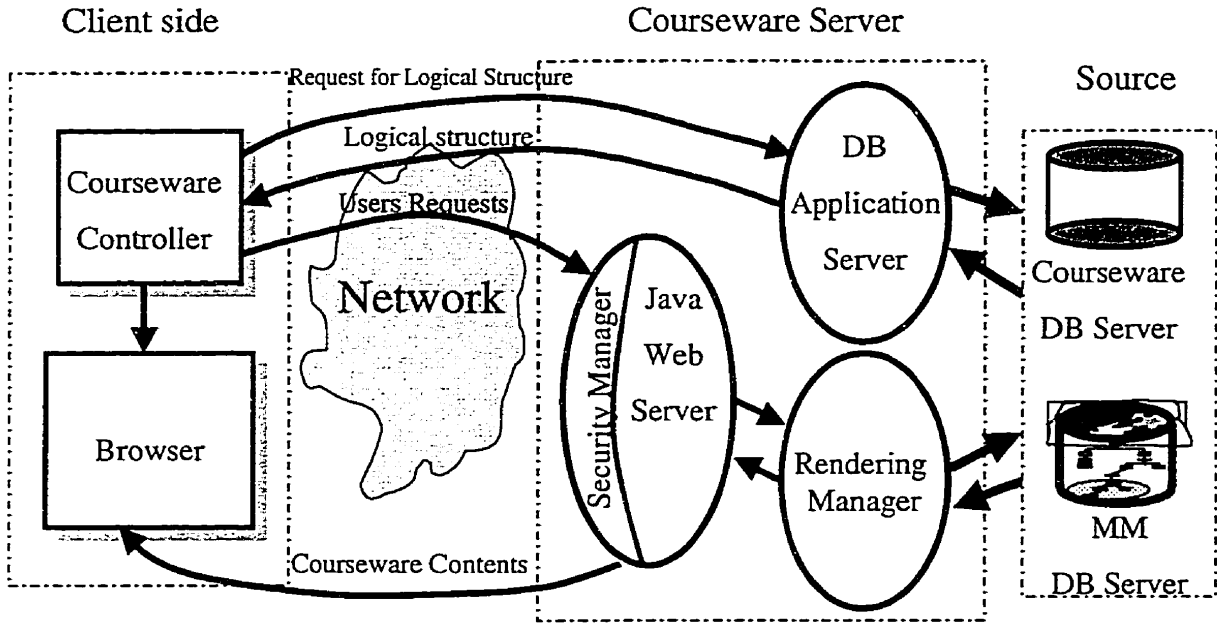


Figure 5.12 MITS System Integration Architecture

In summary, the whole world is moving in the direction of computers and the Internet, and we are in the stage of an information revolution, unparalleled even with the advent of the television and the telephone. Sooner or later like many other areas in life, even educational styles have to update some of their traditions and jump onto the Internet bandwagon if it has to keep pace with changing technologies and life styles [17].



# Chapter 6

## 6 Conclusion

### 6.1 Summary

In summary, distance education is a highway to 21<sup>st</sup> century learning styles, which might be entirely different from those of today. The Multimedia Interactive Telelearning System that is presented in this thesis, aims to build a seamless education environment that supports the delivery of electronic courses to distributed users over the network. The system also aims to combine the learning theory and the emerging technology for facilitating the education process of scattered students. The challenge was to develop a successful Telelearning system that should be available to multiple distributed educational groups on the network regardless of the utilized platforms. Thus, the increased popularity of the Internet and its usage as a communication technology has offered a suitable opportunity to develop such a Telelearning system in order to help students to access courses from any access point at any time.

However, simply putting computers in classrooms, wiring a school, and providing an Internet connection is not sufficient. The effective use of this technology will occur when the educator understands how to integrate it into everyday practice and want to use it. Acceptance of technology in the classroom will be achieved when it is both relevant to educational goals and comfortable to use [45]. Hence, we do not expect that distance education will replace the traditional learning styles, but at least it will obtain a more prevalent position in modern education activities.

We have designed and developed a Telelearning system, keeping in mind the constraints of time, space, location and system compatibility. The strength of the system stems from the proposed courseware data model as well as database schema, the developed database engine, the integrated system's software modules, the generated courseware presentation agents and the courseware reusability.

The courseware data model aimed at describing and representing courseware content in order to facilitate the courseware creation and support the content delivery to distant learners. The produced database schema was designed based on powerful software engineering techniques, which offer the opportunity for future extendibility.

Leading-edge technologies have been utilized to satisfy the implementation of the system schema and the needed database engine modules. Specifically, Java environment and an Object-Oriented database management system have been exploited to accomplish the implementation tasks.

Since the development of the system software components was based on Object-Oriented paradigm and platform-independent programming environment, so the integration of the system modules was achieved smoothly.

## **6.2 Future Work and Suggestions**

Distance education is a dynamic area of research that involves the participation of several disciplines in order to meet the designed goals and satisfy users' needs. Therefore, a lot of beneficial work can be contributed in the area of distance education.

Since the data modeling is one of the most important issues in designing a sophisticated multimedia information system, so a focused attention must be given to the proposed data model. The courseware data model can be modified to include new features at the presentation information level, or can be enhanced to be more effective.

In addition, the Object-Oriented nature of the database schema provides the chance for schema evolution. It is possible to add or modify the state as well as the behavior of the schema classes by inserting or deleting attributes, and updating method definitions. New Object-Oriented software engineering methodologies such as *Unified Modeling Language* “UML” can be used to revisit the analysis and design stages of the system, in order to evolve the schema and enhance the system’s functionalities.

New tools such as Dynamic HTML can be employed to add new interaction features to the system, or to improve the appearance of the rendering application and enhance the courseware presentation. Dynamic HTML allows developers and Web page designers to offer more creativity, control and sophistication to their Web sites [46]. It is based on the Object-Oriented model and extends HTML static nature by allowing scripts or programs to change styles and attributes of page elements, or even to replace existing elements with new ones. It also offers the developers the ability to dynamically change the style, content and structure of Web-based content, while providing them with a detailed control over the appearance, interactivity and multimedia objects required for a modified and exciting application [46].

The system can exploit Dynamic HTML features (e.g. dynamic styles, absolute positioning, dynamic contents and multimedia controls) to enhance the presentation

appearance. First, dynamic styles allow authors to change the size, color or other font properties of a text object within a Courseware-Section. Second, media object position can be changed using absolute positioning features, for instance, moving an image object on top of text object or placing an object in different x,y,z-planes. Thus, students will retrieve a courseware, where its media objects are moving or overlapping using rich multimedia and layout effects. Third, by manipulating object coordinates and other dynamic styles using scripts [46], designers can move media objects around courseware page, thus animating the page. In addition, courseware content can be modified dynamically on the fly in response to student interaction or author updates. This feature enables designers to insert or delete a media object as well as modify text object properties using script languages (e.g. Java, Visual Basic and others). Moreover, multimedia controls can be used to apply visual effects to media objects on a Section page or Chapter page or entire Courseware page. These controls support filters, animation and transition of media objects, e.g. audio object can fade in and out to correspond with the characters' movements. However, Dynamic HTML was launched at the fourth quarter of 1997 and needs time to be experimented with.

Although, the current approach of MITS is implemented, based on the client-server architecture, another major area of research is to investigate the adoption of **mobile-agent** architecture in designing and developing an enhanced version of the system, in order to serve a large number of distributed users.

Finally, since Java has been utilized to develop the system, and with the new release of the Java core that contains CORBA APIs, so it is possible to develop a new fully distributed version of the system.

## 7 References

- [1] L. Sherry, "Issues in Distance Learning", *International Journal of Distance Education*, Volume 1, Number 4, September 1996, pp. 337-365.
- [2] R. Wang, and A. Karmouch, "A Broadband Multimedia Telelearning System", *Proceedings of 5<sup>th</sup> Int. IEEE Symposium on HPDCMCE*, August 1996, Syracuse, USA.
- [3] R. Wang, "A Broadband Multimedia Telelearning System ", Master Thesis, June 1996, Ottawa, Canada.
- [4] A. Grace, "Can Multimedia Help People Learn Faster", *Proceedings of the IEEE International Conference on Multimedia Computing and Systems*, May 1995.
- [5] R. C. Schank, "Active Learning through Multimedia", *IEEE Multimedia*, Volume 1, Number 1, Spring 1994, pp. 69-78.
- [6] C. Vieville, "Organising Distance Learning Process thanks to Asynchronous Structured Conversations", *Proceedings of WebNet'97*, November 1997, Toronto, Canada.
- [7] T. Ingebritsen, G. Brown, and J. Pleasants, "Teaching Biology on the Internet", *Proceedings of WebNet'97*, November 1997, Toronto, Canada.
- [8] Office of Technology Assessment, "Power on New Tools for teaching and Learning", US. Congress Library, OTA-SET-379, Washington DC, USA.
- [9] J. Emery, and A. Karmouch, "A Playback Schedule Model for Multimedia Documents", *IEEE Multimedia*, Volume 3, Number 1, Spring 1996, pp50-61.
- [10] N. Hirzalla, O. Megzari, and A. Karmouch, "An Object-Oriented Data Model and A Query Language for Multimedia Database", *IEEE ICECS*, December 1995, Amman, Jordan.
- [11] B. Furht, "Multimedia Systems: An Overview", *IEEE Multimedia*, Volume 1, Number 1, Spring 1994, pp47-59.
- [12] D. Adjeroh, and K. Nwosu, "Multimedia Database Management - Requirements and Issues", *IEEE Multimedia*, Summer 1997, Volume 4, Number 3, pp24-33.
- [13] K. Meyer-Wegener, "Database Management for Multimedia Applications", *Springer-Verlag*, Spring 1994, Berlin, Germany, pp105-119.
- [14] Elmasri R., and Navathe S., "Fundamentals of Database Systems", *the Benjamin Cummins*, 1994, Redwood City, CA., USA.

- [15] Object Design, "ObjectStore Manuals", Object Design, 1996, Boston, USA.
- [16] A. Seffah, and R. Bouchard, "The Intranet as a Cognitive Architecture for Training and Education: Basic Assumptions and Development Issues", *Proceedings of WebNet'97*, November 1997, Toronto, Canada.
- [17] S. Radhskrishnan, and J. Bailey, "Web-Based Educational Media: Issues and Empirical Test of Learning", *Proceedings of WebNet'97*, November 1997, Toronto, Canada.
- [18] J. Rumaugh, and others, "Object-Oriented Modeling and Design", 1991, Englewood Cliffs: Prentice Hall, USA.
- [19] P.O. Brien, "Making Java Objects Persistent", *Object Design* <http://www.odi.com/>, White paper, November 1996, USA.
- [20] C. Chen, D. Meliksetian, M. Chang, and L. Liu, "Design of a Multimedia Object-Oriented DBMS", *Multimedia Systems, Press ACM*, Volume 3, Number 5-6, November 1995, pp 217-227.
- [21] "Netscape", <http://home.netscape.com/>.
- [22] "Internet Explorer", <http://www.microsoft.com/>.
- [23] H. Zhang, S. and Smoliar, "Content-Based Video Indexing and Retrieval", *IEEE Multimedia*, Volume 1, Number 2, August 1994, pp 62-74.
- [24] S. Palacharla, A. Karmouch, and S. Mahmoud, "Design and Implementation of a Real-time Multimedia Presentation System using RTP", *International Proceedings of IEEE COMPSAC '97*, August 1997, Washington DC, USA.
- [25] T. Meyer-Boudnik, and W. Effelsberg, "MHEG Explained", *IEEE Multimedia*, Volume 2, Number 1, Spring 1995, pp 26-38.
- [26] M. Muhlhauser, and J. Gecsei, "Services, Frameworks, and Paradigm for Distributed Multimedia Applications", *IEEE Multimedia*, Volume 3, Number 3, Fall 1996, pp 48-61.
- [27] H. Khalfallah, and A. Karmouch, "An Architecture and a Data Model for Integrated Multimedia Documents and Presentational Applications", *Multimedia Systems, ACM Press*, Volume 3, Number 5-6, November 1995, pp 238-250.
- [28] N. Poon, and A. Karmouch, "3 Dimensional Multimedia Interactive Courseware Authoring in Telelearning System", *Proceedings of CCECE'97*, May 1997, St. John, Canada.

- [29] A. Karmouch, "A Multimedia Information and Communication System: MEDIABASE," *Proceedings of the ICCM Multimedia Communications' 93 Conference*, April 1993, Banff, Alberta, Canada.
- [30] B. Falchuk, and A. Karmouch, "A Multimedia News Delivery System Over an ATM Network", *Proceedings of IEEE International Conference on Multimedia Computing & Systems*, May 1995, Washington D.C, USA.
- [31] Z.Zhang, and A. Karmouch, "Multimedia Internet Platform for Distance Learning Applications", *Proceedings of 19th Biennial Symposium on Communications*, May 1998, Kingston, Canada.
- [32] K. Nwosu, B. Thuraisingham, and P. Berra, "Multimedia Database Systems – A New Frontier", *IEEE Multimedia*, Summer 1997, Volume 4, Number 3, pp 21-23.
- [33] A. Ghafoor, " Special Issue on Multimedia Database Systems", *Multimedia Systems, ACM Press*, Volume 3, Number 5-6, November 1995, pp 179-181.
- [34] T. Shih, and R. Davis, " IMMPS: A Multimedia Presentation Design System", *IEEE Multimedia*, Spring 1997, Volume 4, Number 2, pp 67-78.
- [35] P. Pazandak, and J. Srivastava, "Evaluating Object DBMSs for Multimedia", *IEEE Multimedia*, Summer 1997, Volume 4, Number 3, pp 34-49.
- [36] B. Rao, "Object-Oriented Database Technology Applications and Products", *McGraw-Hill.*, 1994, New York, USA.
- [37] M. Loomis, "Client-Server Architecture", *Journal of Object Oriented Programming*, Volume 4, Number 9, February 1992, pp75-79.
- [38] I. Jacobson, and others, " Object-Oriented Software Engineering", *Addison-Wesley*, 1993, Englewood Cliffs: Prentice Hall, USA.
- [39] "JavaSun", <http://java.sun.com/>.
- [40] "ObjectStore" <http://www.odi.com/>.
- [41] "JavaSun", "The Java Language An Overview", *Java Sun*, <http://java.sun.com/docs/overviews/java/>, White paper, 1996, USA.
- [42] E. Anuff, "Java Source-Book", *Wiley Computer Publishing*, 1996, NewYork, USA.
- [43] P. Hakkinen, "Possibilities of Java Technology is Distance Learning", White paper, <http://matwww.ee.tut.fi/kamu/distancedocs/pasih.html/>, 1997.
- [44] D. Kramer, "The Java Platform", *Java Sun*, White paper, <http://java.sun.com/docs/overviews/java/>, 1996, USA.

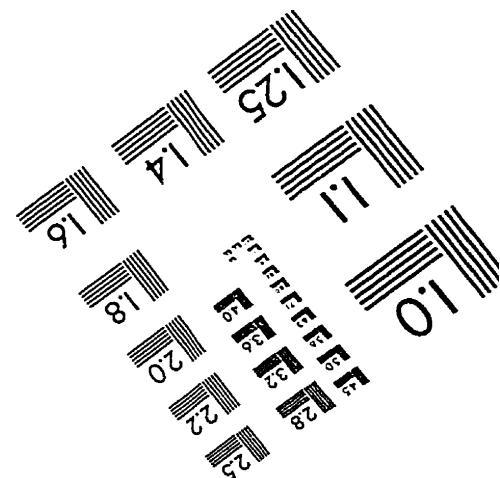
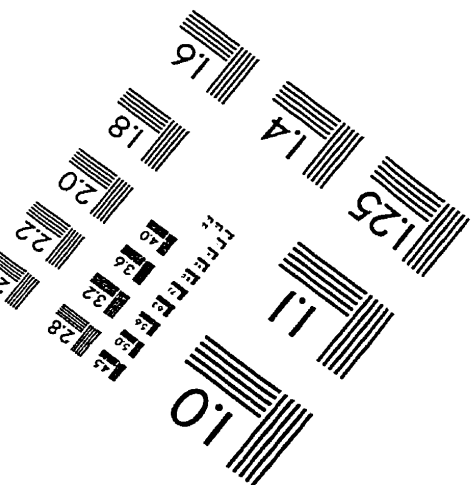
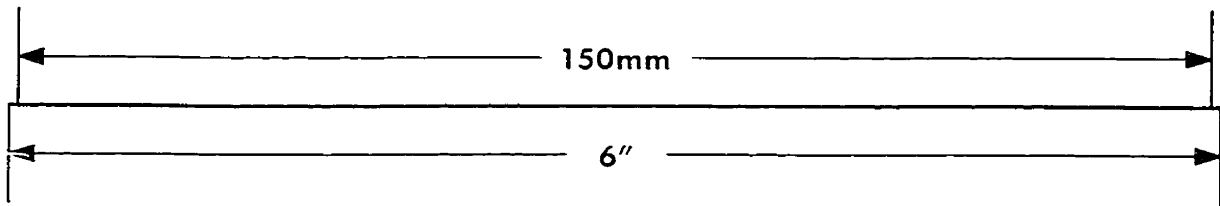
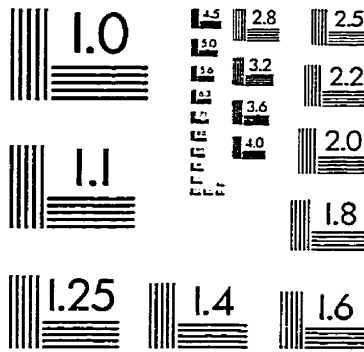
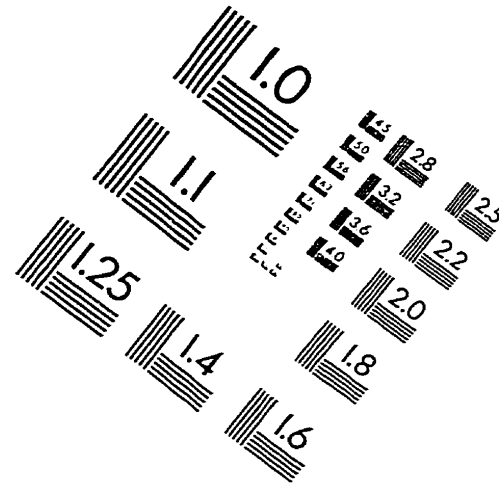
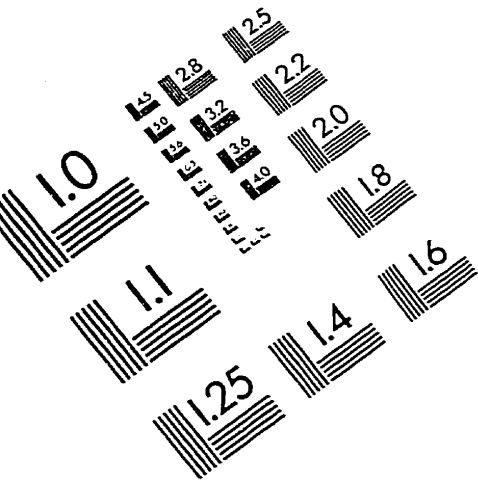
- [45] S. Taylor, and D. Mckay, "The Online Learning Academy", *Proceedings of WebNet'97*, November 1997, Toronto, Canada.
- [46] K. Stremel, and S. Lindsey, "Dynamic HTML: The Next Generation of User Interface Design Using HTML", *Microsoft Corporation* <http://www.microsoft.com/>, White paper, February 1997.



## 8 Publications:

- 1- A. Al-Shammari, and A. Karmouch, "Designing and Modeling a Multimedia TeleLearning Database", *Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering '98*, Waterloo, Canada, May 24-28, 1998.
  
- 2- A. Al-Shammari, and A. Karmouch, "Multimedia Interactive Telelearning Prototype", *Proceedings of International Conference on Computers and Advanced Technology in Education (CATE'98)*, Cancun, Mexico, May 27-30, 1998.
  
- 3- A. Al-Shammari, and A. Karmouch, "On-Demand Multimedia Courseware Delivery over the Network", *Proceedings of INDC'98 - 7th IFIP/ICCC Conference on Information Networks and Data Communications*, Portugal, June 15-17, 1998.

# IMAGE EVALUATION TEST TARGET (QA-3)



APPLIED IMAGE, Inc  
1653 East Main Street  
Rochester, NY 14609 USA  
Phone: 716/482-0300  
Fax: 716/288-5989

© 1993, Applied Image, Inc., All Rights Reserved