INFORMATION TO USERS

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

UMI®

Bell & Howell Information and Learning
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
800-521-0600

Université d'Ottawa · University of Ottawa

Design and Implementation of a Multimedia Interactive Courseware Rendering System

By
Zhongyao Zhang

A thesis submitted to the
School of Graduate Studies and Research
In partial fulfillment of the requirements for the degree of

Master of Applied Science

In

Electrical Engineering

Ottawa-Carleton Institute of Electrical Engineering

School of Information Technology & Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario

July, 1998

Contents

Table of Figures

Abstract

It is anticipated that learning and training will become a dominant sector of the global economy within the next decade. The traditional education system is becoming more and more inefficient due to its limitations, such as passive educational environment, limitations of large classes and accessibility.

TeleLearning system posses many of the attributes required to solve these problems. TeleLearning system shifts the learners from a passive learning environment to a active learning environment. The learners become the center of the education. Recent advancement in multimedia information processing and the Internet technology allow TeleLearning researchers to simulate real classroom on the Internet, and provide users with access to dynamic multimedia content in a virtual classroom at a time convenient to them.

This thesis describes a platform-independent, client/server model for multimedia interactive courseware rendering system. This system helps learners to access course material stored in our multimedia OO database from any access point on the Internet at any time convenient to them.

Acknowledgments

I would like to express my appreciation to my supervisor, Dr. Ahmed Karmouch, for his continuous guidance and patience, for assigning to me such an interesting and challenging research topic. Dr. Karmouch provided a comfortable and well-equipped environment for my work. His work ethics, experience, and encouragement which were the most important factors that helped me finished my M. Sc. program.

I would like to thank all of my colleagues at the Multimedia and Mobile Agent Research Laboratory. Their friendship and kindness made my stay at this lab an enjoyable and profitable one. Thanks also go to Mr. Jianmin Liu for his technical support.

I would also like to acknowledge the support and sponsorship of the TeleLearning Network of Centres of Excellence who provided the necessary funds for this project.

My special thanks were due to my wife, Xizhen, for all of the encouragement received throughout the program.

Acronyms

MITS	Multimedia Interactive TeleLearning System
MICR	Multimedia Interactive Courseware Rendering
CWC	Courseware Controller
CRM	Courseware Rendering Manager
GUI	Graphical User Interface
JVM	Java Virtual Machine
HTTP	Hyper Text Transfer Protocol
OO	Object-Oriented
HTML	Hyper Text Markup Language
VRML	Virtual Reality Modeling Language
URL	Universal Resource Locator
RMI	Remote Method Invocation
CORBA	Common Object Request Broker Architecture

Chapter 1
Introduction

1.1 Motivation

It is anticipated that learning and training will become a dominant sector of the global economy within the next decade. Lifelong learning will be essential for participation in the emerging knowledge economy. [Roxin, 1998]

The traditional education model, namely, "Class-and-Book" model has been efficient for a long period of time. However, this model is becoming more and more inefficient due to its limitations, such as, passive educational environment, limitations of large classes, accessibility for those who have problems joining a class at a specific time, and so on.

All those problems can be solved by TeleLearning systems, which allow knowledge to be stored and transferred through telecommunication technologies. Distance learners can use their computer to take a course through network at any time convenient to them from their home, or their offices. With the help of the Internet, TeleLearning extends access and brings high quality education to all people, regardless of their location, age, or status. Another big advantage of TeleLearning is its interactivity. The interactive component in TeleLearning courseware allows students to control their exploration of new materials in many more ways than the predominantly passive reading of text and reference books. [Sally, 1998] This will change the role of students from passive learning to active learning.

The Internet and multimedia are technologies which have been around for a few years and both are expanding in terms of use and technological advancement. They help to make TeleLearning more and more popular. Multimedia is defined as a computer-based method of presenting information by using more than one medium of communication, such as text, image, video, and audio. Multimedia can provide a variety of visualization aids to better match the different cognitive styles of the students and provides a much richer learning experience than a "One size fits all" textbook. [Sally, 1998] Therefore, TeleLearning applications must supports mechanisms for efficiently handling multimedia data.

Nowadays, more and more people have access to the Internet. It is becoming more of a challenge for TeleLearning courseware developers to make efficient use of the Internet.

As we probably know, the Internet is a worldwide system of linked computer networks with different operating system, such as Unix, Microsoft Windows, Macintosh, etc. It would unrealistic to assume that all users will migrate to a specific platform just to use our TeleLearning system. Therefore, platform independence becomes a big issue for TeleLearning applications designed for the Internet. A successful TeleLearning courseware should be platform-independent in order to be accessed from any point on the Internet. The recent development of Java language allows the potential development of Courseware that can be used on any platform under any operating system.

1.2 Objectives

The main objective of this thesis was to develop a cross platform, multimedia, interactive, courseware rendering system for TeleLearning to help distance learners to access courses stored in our multimedia, object-oriented database. The system is a Java-based, client-server model application, and had to be designed based on Object-Oriented design principles. Especially, emphasis will be placed on the design of the system architecture. Once completed, this architecture will be used to implement the prototype of Multimedia Interactive Courseware Rendering System (MICR) within the University of Ottawa's Multimedia and Mobile Agent Research Laboratory.

The system was designed to operate within the Internet. The new Object-Oriented programming language JAVA™ was exploited to establish platform independence.

1.3 Thesis outlines

The rest of the thesis is structured according to the following outline. Chapter 2 introduces the Multimedia Interactive TeleLearning System (MITS) developed in our laboratory. The generic system architecture of MITS is described, the common requirements of MITS are identified. Roles of Five major components of MITS, namely, Media Production Center, Courseware Author, Online Facilitator, Courseware Database Server and Courseware Rendering System, are briefly introduced. MICR is a subsystem of MITS responsible for courseware rendering.

Chapter 3 will briefly present Java, a new approach to programming for the Internet. Both advantages and restrictions of the Java language are examined. Chapter 3 also provides the comparison of Java Servlet which is the server side Java application with the traditional server application CGI.

Chapter 4 discusses the design of MICR system. In this chapter, the design issues, system architecture and major components of MICR, namely, the front-end application Courseware Controller, the middleware application Multimedia Courseware Server, and the back-end database are described.

Chapter 5 gives the implementation details of MICR. The integration with ObjectStore is also described. The thesis will conclude with chapter 6, which summarizes the conclusions of this thesis and provides suggestions for future research activities.

1.4 Main Contributions

The main contribution of this research is to design and implement a Multimedia Interactive Courseware Rendering System (MICR) for TeleLearning which is platform independent, robust and scalable. With this system, distance learners can access courses stored in our object-oriented database from any point on the Internet with Java enabled web browsers. The first aspect of this research is the development of the system architecture based on three-tier database communication model and a set of fundamental requirements specified in this thesis. Special emphasis is placed on designing a generic client-server model for database applications through the Internet. This model allows different types of databases to be integrated in our system without changing the front-end application. Another aspect is the implementation of the MICR system based on the above system architecture using JAVA™ programming language. A set of generic Java classes that form the infrastructure of any collaboration environment was created and can be reused to implement other client-server applications.

There is one research presentation resulting from this thesis. At the Second Annual Conference of TeleLearning Network of Centres of Excellence (TeleLearning NCE), November 4-6, 1997 in Tornoto, we had a demonstration booth where people could come to interact with the system, and read our sample courses stored in our database located in our laboratory. In addition, the prototype of MICR system was exhibited at OCRI'98 Ottawa Technology Showcase. Two papers arose from this research, one submitted to

the *IEEE Canadian Conference on Electrical and Computer Engineering, 1998,* and another one for *19th Biennial Symposium on Communications 1998*

1.5 Publications arising from the research

Zhongyao Zhang, Ahmed Karmouch *Multimedia Courseware Delivery over the Internet* Proceedings of IEEE Canadian Conference on Electrical and Computer Engineering' 98, Waterloo, Canada, May 24-28, 1998

Zhongyao Zhang, Ahmed Karmouch *Multimedia Internet platform for Distance Learning Applications* 19th Biennial Symposium on Communications Kingston, Canada, May 31 to June 3, 1998.

Chapter 2
Multimedia Interactive TeleLearning System (MITS)

2.1 Introduction

Using computers and telecommunication technologies for delivering education has been a subject of interest for many years. The filed of TeleLearning, which allows knowledge to be stored and transferred through telecommunication technologies, is growing rapidly. This makes TeleLearning a viable alternative to traditional education.

This chapter describes a brief history of TeleLearning, and explains why TeleLearning system becomes more and more popular with the development of computer and telecommunication technologies. MITS is a successful TeleLearning system designed

and developed in the Multimedia and Mobile Agent Research Laboratory at the University of Ottawa. The requirements, and generic architecture of MITS are examined in this chapter.

2.1.1 TeleLearning: History

The term "TeleLearning" or "Distance Learning" has been applied interchangeably by many different researchers to a great variety of programs, providers, audiences, and media. Its hallmarks are the separation of teacher and learner in space and/or time, the volitional control of learning by the student rather than the distance instructor, and noncontiguous communication between student and teacher, mediated by print or some form of technology. [Sherry, 1996]

We find a rich history as each form of instructional media evolved, from print, to instructional television, to current interactive technologies. Traditionally, knowledge is transferred to learners linearly in the form of text book [Gu 1995]. This was the accepted form until the middle of this century, when instructional radio and television become popular. The major drawback of radio and broadcast television for instruction was the lack of a 2-way communication channels between teacher and student. The experience of the Open University in the United Kingdom indicates that the communication between teacher and student makes a profound difference. As increasingly sophisticated interactive communication technologies became available, they were adopted by TeleLearning researchers. Using computers and telecommunication technologies for

delivering education has been a major goal for TeleLearning research. Currently, the most popular media are computer-based communication including electronic mail (E-mail), bulletin board system (BBSs) and Internet, telephone-based audio-conferencing and video-conferencing. The video- and audio-conferencing will not be covered in this thesis. They are beyond the scope of this research. Netscape Browser, Internet Explorer, which are graphical user interface to the World Wide Web, have become popular all over the world. More and more people buy Internet connections for their home computers. It is a right direction to develop computer-based TeleLearning applications to simulate a virtual classroom where users from distance locations can access course material through the Internet. Since users connected to the Internet may use different operating systems or platforms, developing platform-independent TeleLearning applications is a big challenge for TeleLearning researchers. With the advent of the Java programming language, it is practical to develop efficient and platform-independent TeleLearning application to help people to get knowledge from anywhere at any time.

2.1.2 Why TeleLearning

Imagine that you are a middle-aged person with a family and a high-paying job. You also live in a rural city far away from any type of college or university where you desire to get a second degree or a master's degree. You do not want to move your entire family and lose your job just so you can go back to school. Again, imagine that you are active in the military and you are trying to get a degree at a university, but you can not seem to get anywhere because you are transferred from base to base and your credits from one

university don't completely transfer to another university. These problems and many like these have been solved by distance education: TeleLearning [Eggebraaten, 1997].

Distance education is defined as the existence of a time distance and/or a physical distance between the student and the professor, with a form of telecommunications technology utilized to bridge the gap [Wang, Thesis].

In the traditional education system, the teacher controls the whole learning process. Students are in very passive positions where they are forced to acquire knowledge page by page. [Eggebraaten, 1997] In TeleLearning process, students are active participants. They affect the manner in which they deal with the material to be learned. They have a sense of ownership of the learning goals.

The TeleLearning program is convenient for many reasons. For one you don't have to take off two years or more of your time to move to a university that offers the program that you want. Another convenient aspect about TeleLearning is that you study when you have time. You don't have any commitments such as you would in an on-campus sitting. [Eggebraaten, 1997]

Is TeleLearning the way of the future? Many students and instructors believe that distance education is definitely the way of the future because of the convenience and flexibility that it offers to education. However, TeleLearning will never completely replace traditional education. The biggest disadvantage to taking a TeleLearning course

from the students' perspective is the lack of face-to-face interaction with the instructors and other students. This kind of interaction is quite a important issue faced by TeleLearning researchers. In our prototype, we tried to provide more interactivity to simulate a real classroom at three levels. [Huang, 1998]

2.1.3 Multimedia for TeleLearning

Multimedia is defined as a computer-based method of presenting information by using more than one medium of communication, such as text, graphics, audio, and video, and emphasizing interactivity. Advances in audio and video synchronization allow you to display moving video images within on-screen windows.

Interactive multimedia has opened a new research dimension in the theory of learning by introducing an innovative interface where users can navigate for information in various mode of representation such as text, image, animation, sound and video. [Gu, 1995] With multimedia technology integrated, the computer-based learning system can provide the students a more lively and more expressive world of knowledge.

2.2 Requirement of MITS

This section examines the requirements of MITS. Several aspects have been taken into consideration during MITS's design stage.

2.2.1 TeleLearning—Need for platform Independence

In TeleLearning the learner's groups are usually very heterogeneous. Computer platforms may be different from organization to organization. This creates a challenge to come up with a generic system that would work for every user in spite of the type of platform he/she is using. All needed software should be available to everyone participating the course. However, programming cross-platform TeleLearning application to be run on the client side is impossible using traditional programming. Most of applications implemented using traditional programming language are designed for a specific operating system or computer, or compiled for a few computer platforms. When such kind of application was available, the user had to find the application, choose the right version for his/her machine and operating system, then, install the program and finally run it. With the introduction of a new programming language, Java, design and implementation of reliable, secure, and platform-independent TeleLearning applications becomes possible. By embedding Java applets in HTML documents, the application are now going to the user, instead of the user going to the application as was the case traditionally.

2.2.2 Course on Demand and Learner-Centered Education Environment

In the traditional education system, all courses were offered in the classroom. A student who wants to take one course must go to classroom on a specific location at a specific time. These who are on job, or serve in military, or live in a rural community far away from any type of college or university where they desire to take a course, will not have the opportunity to sit in a classroom at a specific time. The traditional education and training system are not satisfactory enough in accessibility and flexibility.

With the advent of TeleLearning system, all those inconvenience will not exist any more. In TeleLearning system, courses are stored in a course database and offered over network. Students have the flexibility to access the knowledge repositories to take a course from any access point in the network at a time convenient for them. TeleLearning applications will help the student to guide the knowledge diffusion process and tailor the content and the learning speed to his/her own learning style and cognitive ability. [Wang, 1997]

TeleLearning system change education environment from Teacher-Centered environment in traditional education system to Learner-Centered environment in TeleLearning education system. In traditional education system, teachers are the center of the learning process. They organize lectures, assignment, as well as exams for a class of students. In

TeleLearning system, the students become the center of knowledge acquisition process. They are motivated by a clear goal, learn by practicing or browsing intentionally, and finally find out answers or get the skills they are seeking to obtain.

2.2.3 Interactivity in TeleLearning System

Interactivity between teacher and students and among the students themselves must be provided by a successful TeleLearning system. In [Sherry, 1996], Sherry, L. noted that, though students felt that the accessibility of distance learning courses far outweighs the lack of dialogue, there is still a considerable lack of dialogue in tele-courses when compared to face-to-face classes. Interactivity becomes a big issue for TeleLearning researchers. Without interactivity, the student becomes autonomous and isolated, and often drops out of college because of the frustration which can result from a lack of understanding.

Successful TeleLearning systems involves interactivity between teacher and students, between students and the learning environment, and among students themselves, as well as active learning in the classroom. [Sherry, 1996]

Interactivity takes many forms. It is not just limited to audio and video, nor solely to teacher-student interactions. It represents the connectivity the students feel with the distance teacher, and facilitator and their peers.

The term "Interactivity" used in MITS refers to the mutual and reciprocal actions and responses between the end-user (the learner) and the courseware-related entities, such as course material, course instructor or online facilitator. In MITS, interaction was classified into three levels: [Huang, 1998]

- Between the learner and the courseware [L-C]
- Between the learner and the facilitator [L-F]
- Between the learner and the learner [L-L]

Among the three, L-C interaction is the most attractive and effective facility in terms of flexible learning environment. Learners can feel free to control the course of hunting knowledge without the influence of other people. L-F interactivity provides the interactions that are unforeseeable by the course author. The main aim of it is to help the learner solve the imminent problems during studying. L-L interaction is a peer-to-peer interaction that only happens when two or more learners browse the same course at the same time.

2.2.4 Integrated, open and scaleable environment

A TeleLearning system may involves many function subsystems from course creation, storage, and distribution to on-line facilitator. As noticed, computer and telecommunication technologies are developing very fast. Today's impossibility may turn into tomorrow's possibility. A good TeleLearning system should be general enough to include not only various teaching architectures, but also various learning paradigms such as lectures, libraries.[Wang, thesis] The system should be developed modularly and

able to be constructed according to the demands of different courses and students. Communications between the students and the professors should be achieved by means of real-time multimedia conferencing system, email, or telephone.

2.3 Generic System Architecture

A set of high level requirements for MITS has been identified in the previous section. This section decomposes those high level requirements into a set of system functions, and refines them leading to a generic system architecture of MITS.

MITS is a distributed TeleLearning system, which is based on the client-server architecture. It is an innovative TeleLearning system, which uses an object-oriented database for supporting persistence database capabilities, and Java environment for developing platform-independent system function modules. A courseware data model has been proposed by [Shammari, 1998] to describe and represent courseware content in order to facilitate courseware authoring and to ensure its delivery to distant learners. MITS consists of five basic components distinguished by the services they offer. A generic architecture is depicted in Figure 2.1.

- **Courseware Rendering**

Courseware Rendering is a subsystem of MITS. It is a Java based client server Internet application. It provides a tool for user to browse multimedia courses stored in courseware database server from any access point on the Internet. This subsystem

exploits the latest information technology such as Java Applet, Servlet, object-oriented, multimedia database, HTML, and VRML, etc., provides a platform-independent learning environment to users. Through the interaction mechanism provided in this system, learners can control the learning speed and present the content according to their own preference.

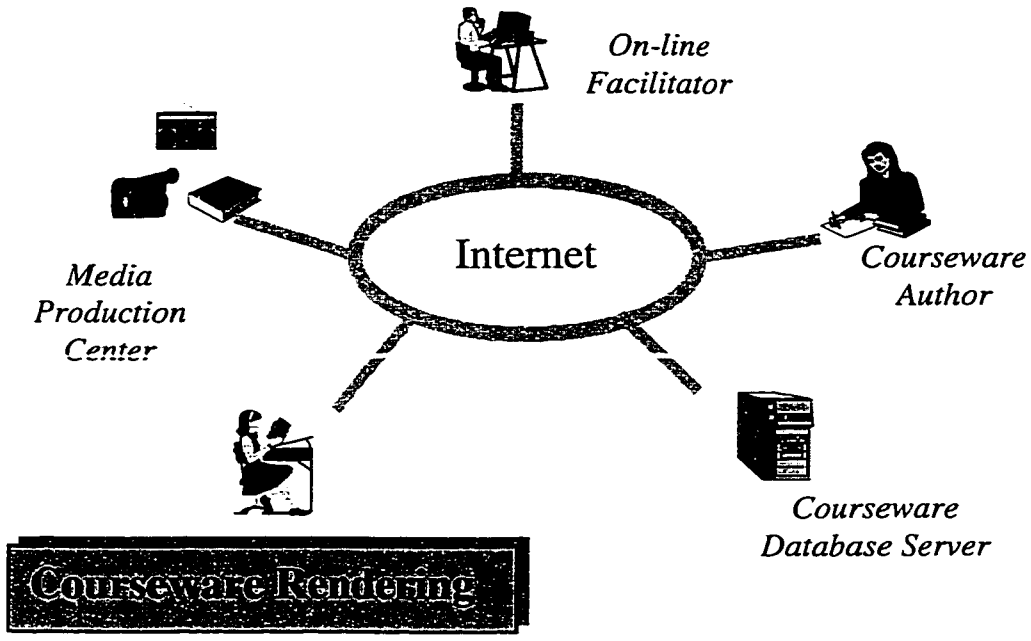


Figure 2.1 MITS generic architecture

- **Courseware Database Server**

The courseware database server is a commercial Object-oriented database management system from Object Design called ObjectStore. It is responsible for storing and managing courseware objects.

Courseware content were prepared by courseware author using Courseware Authoring tools based on courseware data model. Finished courses are stored and managed by

object-orient database: ObjectStore. The Rendering Agent will access the database server on behalf of the learner distributed over the Internet. As mentioned before, the Rendering Agent is a Java Applet shared by users to communicate with courseware database server. Because of the security restrictions of Java, applets are allowed to establish network connections only to the machine from which they are downloaded. This issue forces us to use a central server architecture. The central server was implemented as multi-threaded, stand-alone Java application using core Java package and ObjectStore Java API. (ObjectStore 5.0)

- **Courseware Authors**

Courseware author is responsible for constructing courses based on courseware data model. Courseware authoring involves three major procedures: [Poon, 1997]

- The searching for course material
- The organizing of course material
- The integration of media objects

A courseware author first browses the courseware database for the reuse of courseware media object. If a required media object does not exist, the media production center will be used to acquire the media. After obtaining all the courseware material, they can be organized in two steps: the logical arrangement of courseware topic, and the grouping of related media objects to each topic. The integration of media object in an interactive multimedia document including the building of spatial structure, temporal structure and behavior structure. The spatial structure in a multimedia document defines the layout of media objects within the display area. The temporal structure describes the rendering

behavior of media objects through time. A behavior structure is required to describe how an object should react to user interactions. A courseware is stored in an object-oriented database after being created, and it can be updated in both the content and the scenario (temporal and behavior structure) at any time.

- **On-line facilitators**

On-line facilitator was designed to establish the communication between the teacher and student. Teacher or specialists will work on-line to answer questions or discuss interesting topics with students on request.

- **Media production center**

Media production center is responsible for capturing information from the real world and coding them into different media objects such as text, image, audio, video. These media objects are stored in database as well, and used as basic material for courseware authoring.

Chapter 3
Java: Applet and Servlet on the Internet

3.1 Introduction

As pointed out in chapter 2, in TeleLearning the learners' group are usually very heterogeneous. Computer platforms used by users are always different, too. A good TeleLearning system should be platform-independent in order that it is accessible to the majority of users regardless of what platforms they are using. With traditional programming language, developing cross-platform applications is almost impossible. With the advent of a new programming language, Java, it is possible to design and implement architecture neutral applications. Automatic garbage collection, objects, threads, networking, exception processing, no pointers and a great initial set of classes, all combine to make Java a truly wonderful system. Because of advantages of Java over

current languages that makes it superior for Internet programming. It was decided to use Java applets as the standard for writing TeleLearning applications.

We also mentioned that multimedia courses are stored in multimedia object-oriented database as course objects, and will be rendered to user over the Internet as requested. There is a need for an interface in order to present multimedia objects such as text, image, audio, video. As we promised, our TeleLearning system can be accessed from any point on the Internet. It was decided to use the most basic interface to the Internet: Web browser for rendering our course material. Web browser is a window to the Internet that can present multimedia objects such as text, image, audio, video with the help of helper applications and plug-ins.

Traditionally, the word *Applet* has come to mean any small application. In Java, an applet is any Java program that is launched from a web document, that is, from an HTML (Hyper Text Markup Language) file. Java applet receives a lot of information from the web browser: it needs to know when it is to be initialized, when and where to draw itself in the browser window, and when it is activated or deactivated. A browser that can run Java applet is referred to as a Java-enabled browser. Today, more than 90% of web browsers being used are Java-enabled [Shirmohammadi, 1997].

Servlets, on the other hand, are Java programs designed to execute at the server site. It is server side equivalents of applets. In MITS, course contents are transmitted from the server site to the client site as HTML documents over the Internet. Those HTML

documents are created by the Rendering Manager, which was implemented as a Java servlet, on the fly. Why do we choose Servlet API not CGI (Common Gateway Interface) to implement our server site application to serve dynamic HTML is examined later in this chapter.

In the next section we will describe the advantages and the restrictions of the Java language.

3.2 Java and its restrictions

3.2.1 Characteristics of Java

The name Java is a trademark of Sun Microsystems and refers to the programming language developed by Sun. According to the information provided by Sun Microsystems [Sun, 1997] [JDK, 1997], Java is:

" ... *simple, object-oriented, distributed, interpreted, robust, secure, architecture neutral, portable, high-performance, multi-threaded, and dynamic language.*"

Although the above statement might seem like an advertisement's chain of buzzwords, it truly describes the key technical features of Java.

- **Simple:**

Java is based on the C++ programming language, but removed many of the language features that are rarely used or often used poorly, such as pointer which is one of the most

bug-prone aspects of C/C++ [Glenn, 1997] [Wutka, 1997] [Cassady-dorion, 1997]. This makes Java easy and quick to learn compared to other programming language. C++ is a language for object-oriented programming and offers very powerful features. However, as is the case with many languages designed to have power, some features often cause headache. Multiple inheritance is one of such tricky features in C++ which is complex and often cause problems. Java gets rid of this headache. It does not support multiple inheritance. Instead, Java introduces interfaces. A class in Java can implement one or more interfaces. The benefits of using interface are much the same as the benefits of using abstract classes. Interfaces provide a means to define the protocols for a class without worrying with the implementation details. In addition, Java has an automatic system for allocating and freeing memory (garbage collection), so it is unnecessary to use memory allocation and de-allocation functions as in C and C++.

- **Object- Oriented**

Simply stated, object-oriented design is a technique that focuses design on the data (objects) and on the interfaces to it. Ideally, object-oriented design can permit the creation of software components that can be reused. Java is designed to be object oriented from the ground up. Java can be considered as a 99% pure object-oriented language. All program code and data is contained within objects and classes, with exception of simple types such as integers and characters which are implemented outside of the object system for reasons of efficiency. Technically, Java's object-oriented features are those of C++ with extensions from Objective C for dynamic method resolution.

- **Distributed**

Unlike the languages C++ and C, Java is specifically designed to work within a networked environment. Java has a large library of classes for communication using the Internet's TCP/IP protocol suite, including protocol such as HTTP and FTP. Java code can manipulate resources via URLs as easily as programmers are used to accessing a local file system using C or C++ [Cassady-dorion, 1997] [Venhelsuwe, 1997].

- **Interpreted**

When the Java compiler translates a Java class source file to bytecodes, this bytecode class file can be run on any machine that runs a Java interpreter or Java-enabled browser. This allows the Java code to be written independently of the users' platforms. Interpretation also eliminates the compile and run cycle for the client because the bytecodes are not specific to a given machine but interpreted. This feature enable a Java application to be executed on any platform, and make the life of TeleLearning application developers easier. Another important feature of Java gained from bytecode is Java's security characteristic. Java virtual machine provides bytecode verifier to check the bytecode fully for things like mismatched parameters, bypassing access restrictions, and so on. [Venhelsuwe, 1997]

- **Multithreaded**

Java is a language that can be used to create applications in which several things happen at once. Multithreading is a way of building application with multiple threads. As we

know, writing programs that deal with many things happening at once can be much more difficult than writing in the conventional single-threaded C and C++ style. Fortunately, Java has a sophisticated set of synchronization primitives for handling threads, including concurrency and consistency [Cassady-dorion, 1997].

- **Network programming**

Java's characteristics make Java a good candidate for network programming. Today, the most likely place you'll find Java is on the Internet. Once your Java applet has been compiled, it can be executed with the Applet viewer, or through any Java-enabled browser such as Netscape browser, Internet Explorer. This can be done by embedding the applet into an HTML document. The HTML file is based on the <applet> tag and takes the following basic structure: [Aunff, 1996]

<HTML>
<applet codebase = *location of code* **code** = *MyApplet.class* **width** = 100 **height** = 150>
<param name = "*parameter*" **value** = "*accepted value*">
</applet>
</HTML>

The <applet> tag contains the filename of your executable code, in the format of *filename.class* , followed by the dimensions of your applet given in pixels. The <param> tag accepts the parameter name and its associated value to initialize the applet.

As a network programming language, Java has a lot of advantages over other traditional programming language. But it has some restrictions also, which are examined in the following section.

3.2.2 Java Restrictions

Because applets can be downloaded over the Web, they are, by nature, not secure. Both Sun and browser manufacturers have placed some restrictions on applets. This security was implemented to help applets gain wide acceptance on the net. Applets are not given full access to the local machine's file system. Applets currently do not have the means to save files out to a user's local machine, and they can not read files off the local machine either. Applets are also restricted from loading dynamic or shared libraries from other programming language. Because those libraries can't be verified and they would require access to the file system.

Restriction placed on applet by Web browser is network connectivity. Applets currently are limited to network connections with their host machines. This means applet can communicate directly back to the Web server that it's downloading from, but it can't contact any other servers on the Net [Shirmohammadi, 1997].

3.3 Servlet and CGI

Servlet is the newest aspect of Java networking. In MITS, servlet API was chosen to implement our Rendering Manager to construct HTML page on the fly. In this section, the attractiveness of servlet and the limitations of CGI are examined.

3.3.1 What is Servlet

Servlets are objects that conform to a specific interface that can be plugged into a Java-based server, intended as a Java-style replacement for Common Gateway Interface (CGI) programs. Servlets are to the server-side what applets are to the client-side. They differ from applets in that they are faceless objects (without graphics or a GUI component). They serve as platform independent, dynamically loadable, pluggable helper bytecode objects on the server side that can be used to dynamically extend server-side functionality. Servlet and applet are compare in table 3.1 [Venhelsuwe, 1997].

Table 3.1 Comparison of Servlets and Applets

Servlet	Applet
Subclass of GenericServlet	Subclass of Applet
Runs in a server	Runs in browser
Must be muitithreaded or thread safe	Generally single thread per applet
No direct user interface	Uses AWT for user interface
If downloaded to server, controlled access to files and network	No access to files and network access back only to serving host
If local to server, full access to files and network	N/a

What does a Java Servlet look like? At a superficial level, a servlet is much like an applet. In an Applet class the behavior is largely determined by a few methods, which are called by the browser. These methods are `init()`, `start()`, `stop()`, `destroy()`, and `paint()`. In a Servlet class, the same basic concept is used, but the particular methods of interested are slightly different.

A servlet does not need to become active or inactive in the way that an applet does. Also, a servlet does not have a GUI of its own. The servlet class, therefore, does not define the `start()`, `stop()`, or `paint()` methods. The main behavior of the servlet is

required in response to a new connection at the server, and the connection results in a call to the `service()` method of the servlet. For more detail, see [Venhelsuwe, 1997] and [Cassady-dorion, 1997].

3.3.2 CGI limitations

When HTML and the World Wide Web first invented, the content of each displayed page was essentially static. Each URL (Unique Resource Locator) referred directly to either a fixed page or a fixed element of a page. CGI allows a URL to contain a basic reference, which is not to a page of HTML, but to a program. Parameters can be passed to the browser to control the execution of the program to create a dynamic HTML page. Although CGI is quite simple to use, both from user's and the server administrator's point of view, it has a number of weakness:[Venhelsuwe, 1997]

Low performance Initiating a new operating system process is necessary to execute the CGI program referenced in the HTTP request. This leads to performance problems. In addition, most CGI programs are written using interpreted languages, such as Unix shell scripts and PERL. This is not a requirement of the CGI specification, but appears to be popular options. Using compiled languages improves speed but tends to raise platform-dependency issues.

Startup time CGI programs run as separate processes, which generally involves significant startup time. This overhead occurs each time the program is invoked. The startup time is compound if you are using an interpreter.

Poor inter-CGI communication Because each invocation of a CGI program starts a separate process, communicating between invocations must usually be done via files, and hence can be quite slow. Communicating between different CGI programs on the same server is similarly cumbersome.

Security Undoubtedly the biggest weakness of CGI is a lack of formal security mechanisms. Without a secure runtime system, web master must closely inspect CGI programs for programming flaws and security loopholes. Dynamic content may often be limited to forms processing, to go beyond this, a safe and secure environment is required. A Java-based solution can provide the needed security.

3.3.3 Attractiveness of Java Servlet

As mentioned, CGI has limitations compared to Java Servlet. What are servlet's advantages? I will describe them briefly in the following:

High performance. Unlike traditional CGI request handlers, servlets do not go away when they finish a request. This eliminates the heavy overhead of starting and external program to serve each request; it enables servlets to practically perform all server operations, including basic file serving.

Platform independence. Like applets, servlets run on any server platform that provides a Java Virtual Machine (JVM) supporting the Servlet API.

Secure and Safe. Servlets can also take advantage of Java's security framework, allowing different levels of security for different servlets. For instance, you could define

a security policy that allowed a servlet to access only certain directories on your file system. In addition, you could limit other Java feature, like network access.

Transport independence. The servlet API has base classes that are independent of the transport protocol used to carry web content and subclasses that are specific to HTTP. Specific versions of the Servlet API can be developed for future web transport protocols. Java's Servlet API has a significant advantage over CGI in terms of performance, transport independence, and security. The differences between servlets and CGI are highlighted in table 3.2 [Venhelsuwe, 1997] [Cassady-dorion, 1997]. The use of servlets for all web server operations makes a very powerful and flexible web platform.

Table 3.2 Comparison of Servlet API and CGI [Venhelsuwe, 1997]

	Servlet	CGI
Language	Java	Any programming language
Invocation	Direct method call	Run program
Parameters	ServletRequest methods	Environment variables
Run time checks	Java Virtual Machine	Language specific
Security	Java Security Manager	Operating system specific

Chapter 4
Design of Multimedia Courseware Rendering System over Internet

In this chapter, we present one of the areas of focus of this thesis: the design of a rendering system for TeleLearning. The implementation that we made of this system, which we will present in the next chapter, has been named "Multimedia Interactive Courseware Rendering System" (MICR).

First of all, we analyze the general requirements of MICR. Secondly, we present the system architecture and functionality, detailing the role of its various components.

4.1 General specifications

MICR is a subsystem of MITS. It was designed as a Java-enabled, platform-independent, learning tool for distance education to help people to get knowledge from anywhere at any time. As a subsystem of MITS, MICR should be subject to the general requirements of MITS discussed in chapter 2, such as, platform-independence, course-on-demand, interactivity, and learner-centered environment. The tool renders the content based on the users' request, and the users can control the learning speed and tailor the contents according to their preference. As an Internet application, it should supports the followings:

- **Scalability**

As we know, the Internet is serving more and more users everyday. The number of private users buying Internet connection for their homes is rising very quickly. It has been predicted that the Internet access will be standards of future households as are television and telephone. [Shirmohammadi, 1997] Any system build with Internet in mind should therefore be scalable. MICR is an Internet application based on client/server model. It can be scaled horizontally or vertically. Horizontal scaling means adding or removing clients with only a slight performance impact. Vertical scaling means migrating to a larger and faster server machine.

- **Flexibility/ open system**

The system should be part of an integrated system, so that it can be combined with the existing or future Internet application. It should neither be too complex nor too specific

as these will prevent integration of the system with other systems, as well as limits the evolution of the system and make it useless in the long run.

- **Generic, simple, robust**

As mentioned above, the purpose of the system is to provide a tool to help people to get knowledge from Internet. It should be simple to use. The computer platforms used by users are always heterogeneous. A successful TeleLearning system should be generic and available to the majority of users on the Internet regardless of the platforms used. It should also be easy-to-use, and robust.

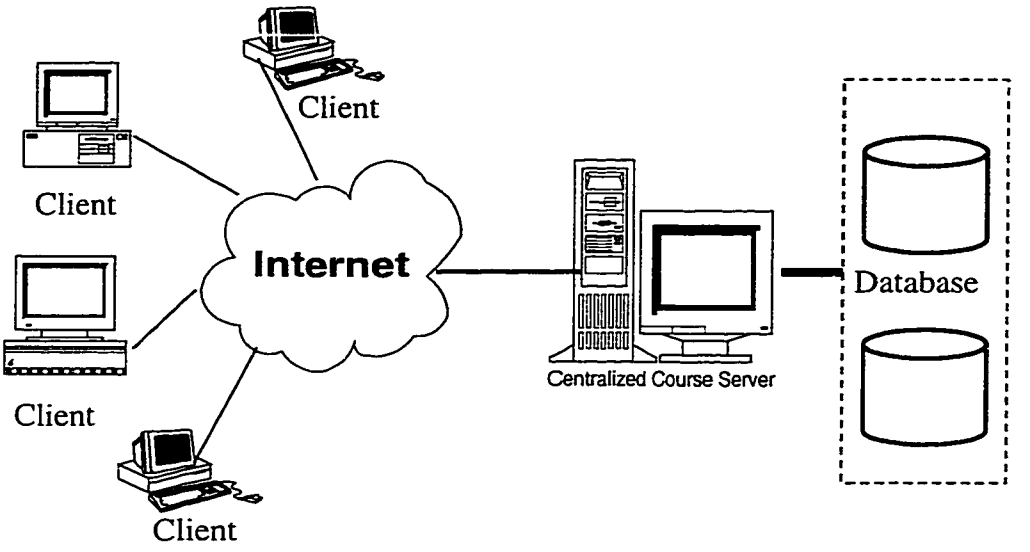


Figure 4.1 Global System Architecture

The system design of MICR is based on client/server paradigm. Because of Java's security restriction, applets can only communicate to the computer from which they are

downloaded. This forces us to choose a centralized server architecture to implement our course server, as shown in Figure 4.1.

Clients distribute over Internet. All requests from clients are handled by the central server. MICR is also based on object-oriented design principles in order to easily incorporate additional modules or derive from existing modules. Incremental development can be achieved, thus making enhancements less cumbersome. With all the requirements in mind, the system architecture and its components are described next.

4.2 System Architecture

4.2.1 Introduction

This section describes the architecture for MICR that meets the system requirements identified earlier in this chapter and chapter 2. Followed by a detailed description of each of the key architectural components. Communication between client and remote objects is also examined by discussing socket programming and Remote Methods Invocation (RMI) which is Java's CORBA.

A client/server architecture for MICR system is depicted in Figure 4.2 and consists of three principal elements: client site, courseware server site, and multimedia courseware database.

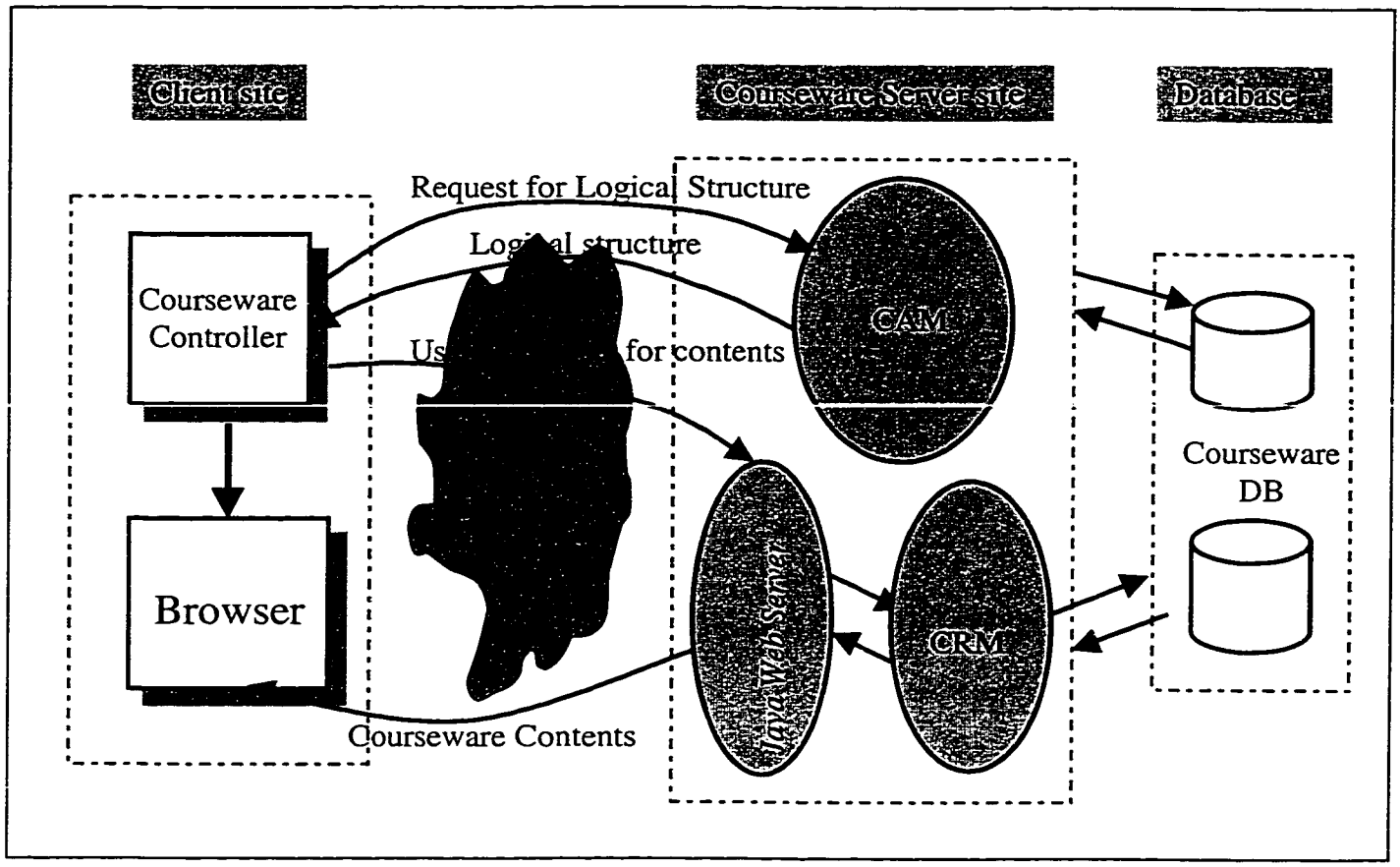


Figure 4.2 System Architecture for MICR

- **Client site**

The client site includes Java-enabled Internet browser such as Netscape browser, Internet Explorer, etc., and the client side application: Courseware Controller (CWC). The CWC is launched from a Java-enabled web browser and communicate with the browser to get parameters for initialization. The CWC is a Java applet shared by users and responsible for establishing a communication channel between the client and the courseware server. The CWC sends user's requests to server, while, server's responses are presented in the user's web browser. Details about the CWC are described later in this chapter.

- **Server site**

The courseware server site is made up of three components: Client Access Manager (CAM), Java Web Server, and Courseware Rendering Manager (CRM). They work together to serve user's request for courseware logical structure and courseware contents.

The CAM is a standalone multi-threaded Java application and responsible for authentication checking and retrieving courseware logical structure from the courseware database. Communications between the CAM and the client are based on TCP/IP protocols.

The CRM is a Java servlet which is a server side equivalent of Java applet. It is managed by the Java Web Server, and is responsible for retrieving course contents requested by the client from the courseware database and constructing HTML pages on the fly based on the user's presentation preferences. Communications between the CRM and the client

are based on HTTP protocols. The course materials may contain various kinds of media objects such as text, image, video, and audio. The real time media objects will be transmitted using Real-time Transport Protocol (RTP).

Java Web Server [JavaSvr, 1998] is a commercial product from Sun Microsystems. It is used to manage our Courseware Rendering Manager. We have selected Java Web Server for the following reasons: [Sun, 1997]

- cross platform: Java Web Server's ability to be cross platform is due to the fact that it is written in Java. Since Java is a widely support and accepted language, most of the platforms support any application written in Java. As a result, Java Web Server can be used on more platforms than any other web server can.
- Servlet API: Java Web Server is the first commercial example of a full-fledged Web server written in Java with "native" support for Java Servlets. The beauty of Java Servlets is that they provide a standard approach to extending server functionality without the limitations of CGI-based or server-specific approaches.

- **Multimedia Courseware Database**

Multimedia Courseware Database (MCD) is our course repository. Created courses are stored and managed by MCD. In our system, we exploited a commercial object-oriented database management system (ObjectStore 5.0 on Java) to generate our database schema [Shammari, 1997], because it offers enhanced tools to manage, manipulate, and maintain complex data such as multimedia information. It also provides rich modeling capabilities.

4.2.2 Design of Courseware Server

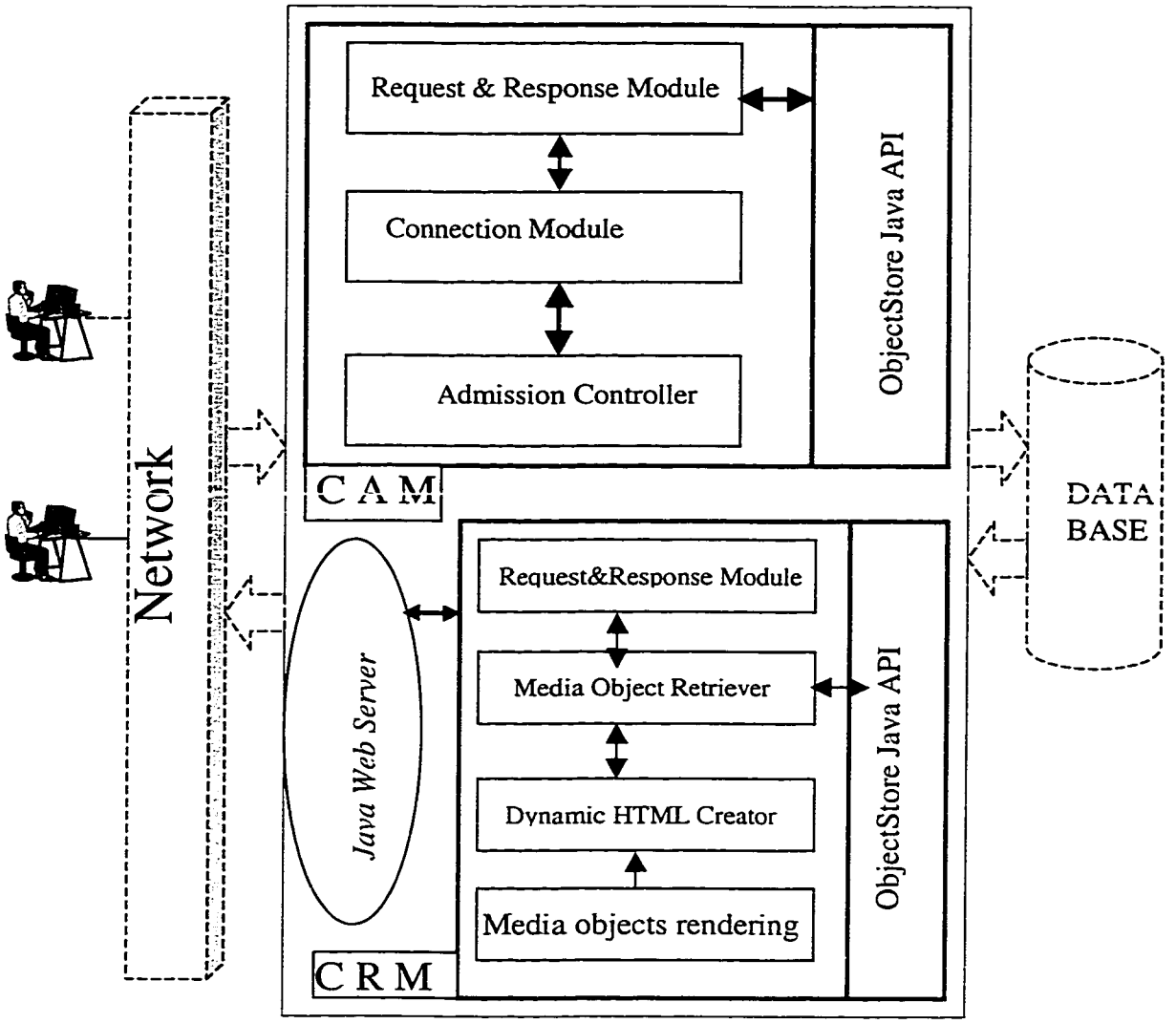


Figure 4.3 Courseware Server Architecture

The main tasks of the Courseware Server are:

- Parsing client's request and construct outgoing stream
- Searching ObjectStore to get course logic structure and contents requested by users.
- Authentication checking
- Creating HTML page on the fly based on the users' presentation preferences.

As we can see from Figure 4.3, the Courseware Server consists of three principal components:

- Client Access Manager (CAM)
- Courseware Rendering Manager (CRM)
- Java Web Server

Each principal components has several functional modules which work together to provide specific service to client. Details about each component are presented next.

4.2.2.1 Client Access Manager

Client Access Manager is responsible for handling user's request for course logical structure and authentication checking.

As shown in Figure 4.3, the Client Access Manager contains the following functional modules: Admission Controller, Connection Manager, and Request & Response Manager. ObjectStore Java API is used as an interface for the CAM to retrieve ObjectStore database.

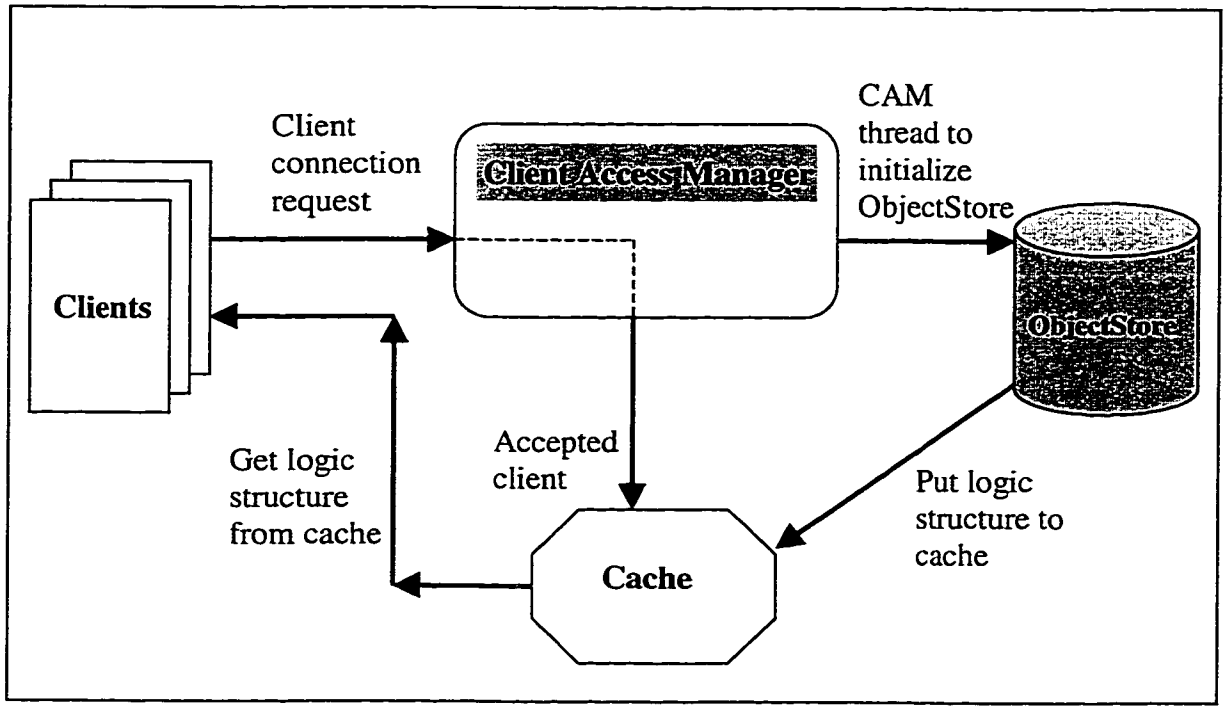


Figure 4.4 Client get logic structure from Cache

When a client request for connection to the Courseware Server, the Admission Controller checks the client's profile file and resource (CPU) availability to decide whether accept or reject the client's connection request. Once the client's connection request is accepted, the Connection Manager creates a new thread for this client and initializes it. After initialization, the thread starts and the Request & Response Manager begins to server the client.

The Request & Response Manager parses request from the client, searches the database, and constructs response stream. The client may request the logical structure for all available courses in the database or courses related to a specific area, for example,

courses about software engineering, or courses about Java language. Two different ways were designed to retrieve the logical structures from the database. Shown in Figure 4.4 and Figure 4.5, respectively.

Figure 4.4 describes the first way to handle users' logical structure requests. As we noticed, it takes time to initialize ObjectStore. Client's thread is not allowed to retrieve ObjectStore, instead, the server (Client Access Manager) implements a thread to initialize the Courseware Database (ObjectStore), and starts a read-only transaction to retrieve the database for logical structure and save the results in cache, after that the thread stops.(see Figure 4.4) Later on, when a client thread starts, it does not need to repeat the job that the thread in the Client Access Manager did to initialize database, and to start transaction. It gets the logical structure from cache instead of from database. The advantage of this method is that the client can get response quickly, because the client thread saves time on initializing ObjectStore. The problem arises from this method is that a big cache will be needed when database becomes big. It is not necessary to put logical structures for all courses in database into cache without caring about the client's need. The problem can be solved by the following method (see Figure 4.5).

Figure 4.5 shows the second way to deal with users' logical structure request. As we know, ObjectStore is thread safe. It allows multiple threads to access the database concurrently. The server allows each client thread to access the courseware database to get available course logical structure from it. In this case, all client threads have the same priority in term of CPU usage. The advantage for this method is that the clients can

retrieve the database to get courses related to their need rather than all courses in database. When the server operates ObjectStore database, ObjectStore's Java API is used.

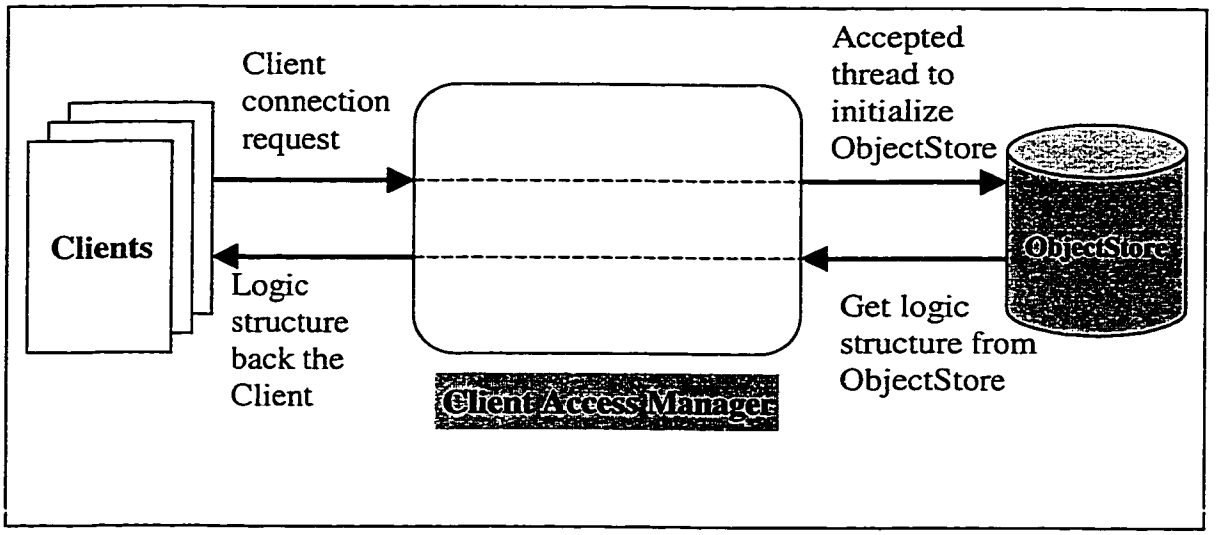


Figure 4.5 Client get logic structure from ObjectStore directly

4.2.2.2 Courseware Rendering Manager (CRM)

As previously described, the Courseware Rendering Manager is responsible for handling client's request for course contents. The main tasks of the CRM are described as follows:

- Parsing client's request
- Retrieving ObjectStore for course contents
- Constructing HTML page on the fly based on client's presentation preference

The CRM was implemented as a Java in a well-defined environment. A commercial product: Java Web Server from Sun Microsystems has been chosen as the environment.

As shown in Figure 4.3, the Courseware Rendering Manager is composed of the following functional modules:

- Request & Response Module
- Media Object Retriever
- Dynamic HTML Creator
- Media Objects Rendering Tools

When the courseware Controller starts to work, the user may select an interested course to learn through the controller, also the user can decide what kind of media object could be included in presentation based on the user's hardware ability. This kind of request is sent to the Courseware Rendering Manager through the Java Web Server over the Internet. The Request & Response Module in the CRM listens and parses the requests from the Java Web Server. The request from the user may contains which part of the selected course he/she wants to read and what kind of media object, such as image, video, audio, and VRML virtual world, will be included. After that, the module informs the Media Object Retriever to get the requested media objects from ObjectStore, calls the Dynamic HTML Creator to construct HTML documents on the fly, and sends the results back to the user's browser over the Internet. HTTP protocols is used to establish communications between the client and the Rendering Manager.

The Media Object Retriever is the interface to the Multimedia Courseware Database (ObjectStore). It operates database through ObjectStore's API, gets the media objects requested by the user.

The Dynamic HTML creator constructs HTML pages dynamically according to user's presentation preference. During the construction of HTML pages, Media Object Rendering Tools will be called. Media Object Rendering Tools are a group of Java Applets to render media objects such as images and audio, and provide users with some interactions such as click, double click, drag and drop, etc.. Video objects can also be rendered with the help of plug-ins.

Next, the client architecture is presented.

4.2.3 Design of MICR Client

MICR client distributed over the Internet. The general layout of MICR client is shown in the following figure: Figure 4.6

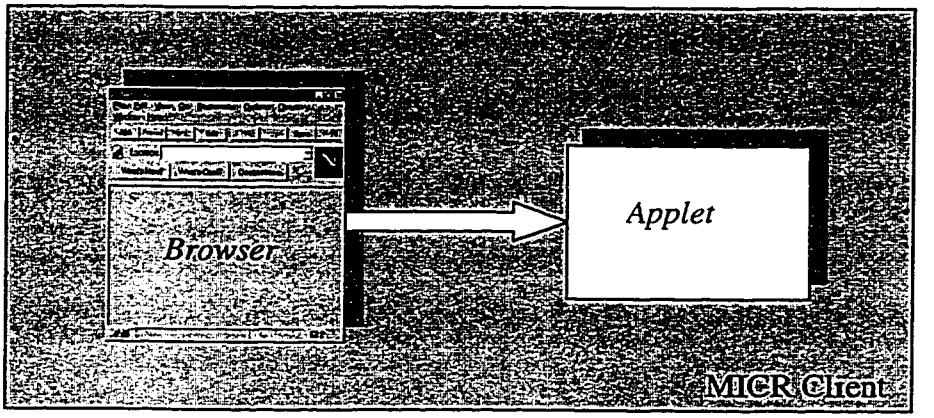


Figure 4.6 Context diagram of MICR client

As previously described, the MICR client is comprised of a Java-enabled web browser and a Java applet which is Courseware Controller (CWC). The CWC is embeded in HTML document and initialized by parameters from the browser.

The main tasks of MICR client are:

- CWC gets parameters such as hostname, port No. from the browser to establish a socket connection to the server.
- Accepts user's identification information and sends to the Courseware Server for authentication checking.
- Reconstructs course logical structure by parsing response data stream from the Courseware Server
- Interprets user's interaction and sends user's requests to the Courseware Server.
- Displays course contents requested by the user in the browser in HTML document format.
- Close connection to server

When an applet is launched in a Java-enabled web browser, the applet needs to get some parameters, such as hostname, port number, etc. from the browser in order to make a connection to the server from which the applet is downloaded. As we probably know, applet is able to find the server name from which it was downloaded. Why should we bother to have a parameter for specifying server name? Specifying server name causes re-writing of HTML file when the server machine is changed and creates inconvenience. Because of security reasons, some computers are protected by a firewall and are

connected to the Internet through a proxy server. The web browser of a client that uses a proxy server thinks the applet's host is the proxy server and not the real server. So it must be told specifically to request connection the applet's server and not the proxy server.

After the initialization of the applet, a connection between the user and the Courseware Server is established. Request for the logical structure for all courseware stored in courseware database is sent to the server, and the MICR is ready for the user to logon. The user must provide identification information such as user name, password for authentication checking. Once passed, the Courseware Controller panel pops up for authorized user to control the learning process. The Controller reconstructs course logical structure by parsing the server's respond data stream for logical structure's request. The titles for all available courses from courseware database are listed on the controller panel for the user to select. Short description for each course will be a help for the user to decide which course to read. During the learning process, the controller interprets the user's interaction, such as which course, which chapter of this course, and what kind of media objects could be included in the presentation, and send those requests to the courseware server for content. The content will be displayed in the user's browser. When the user terminates the learning process, the controller closes the connection to the courseware server.

Currently, communications between the client and the Courseware server was established by socket programming that is a traditional method in distributed computing. With the development of Java language, more new features are available for distributed

computing, such as Remote Method Invocation (RMI), which is a Java's CORBA. Communications between client and remote objects can be achieved efficiently by using RMI or CORBA.

4.2.3.1 GUI Design

The layout of the Courseware Controller graphic user interface is shown in Figure 4.7. As we can see in the main window layout, the main window is split into three main areas: control panel area, display area, and menu bar area. As previously described, request for course logical structure from course database is sent out during the initialization of the applet. After the initialization of the applet, list of available courses in database is displayed in the Display area. The Display area was implemented as card metaphere. First card, as seen in Figure 4.5, consists of two parts: Course List area and Short Description area. Course list is displayed in the Course List area. Short Description area is used for displaying short description for user to review, when the user points his mouse on a specific course. Course short description help the user to decide whether or not this is a course satisfies his requirement. Once the user clicks on one course, second card will show up to display logical structure of the selected course.

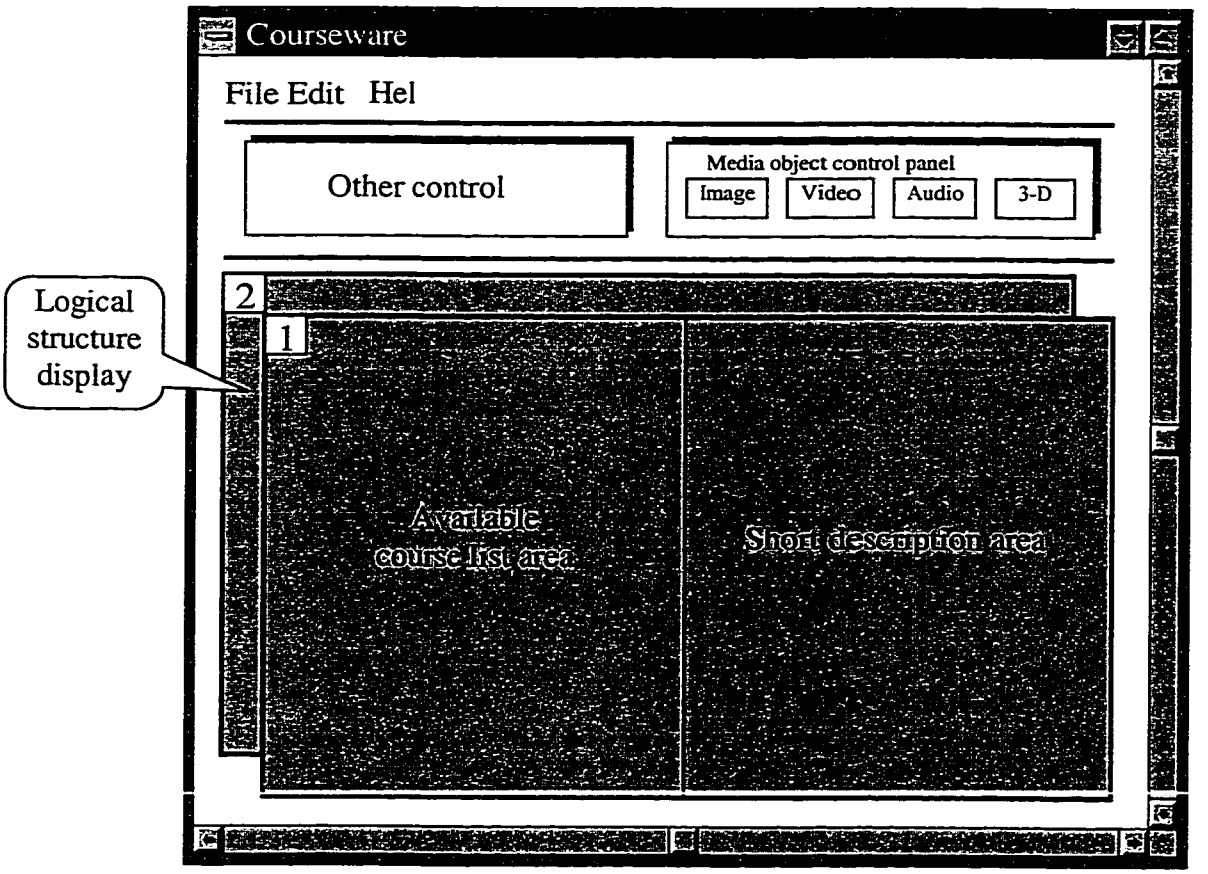


Figure 4.7 Courseware Controller graphic user interface

As we know, video, audio objects are usually big and it takes time to transmit them over network. We provided a media object control panel for the user to control the presentation format. The user can use the control panel to tell the Courseware Rendering Manager which kind of media object will be included in HTML document. As shown in Figure 4.7, the control panel area consists of several self-explanation buttons. The left four toggle buttons are Audio button, Video Button, Image button, and 3-D world button. User can use these buttons to interact with course materials. When a button is pressed down, corresponding media type will be included in presentation. Otherwise, the media

will not be included. When the user click on a section, presentation request together with presentation preference are sent to the Courseware Rendering Manager through the Java Web Server.

4.3 Communication between client and database

In MICR system, the database is used to collect and manipulate our course objects. As noted previously, ObjectStore from Object Design is used as our database server. ObjectStore emphasizes on client/server architecture and offers full database support for multiple clients distributed over the network. During our system design stage, two database communication model were compared: Two-tier database communication model, and Three-tier database communication model.

4.3.1 Two-tier Database Communication Model

Two-tier models appeared with the advent of server technology. Communication protocol development and extensive use of local- and wide-area networks allow developers to create an application front end that accessed data through a connection (socket) to the back-end server. Figure 4.8 illustrates a two-tier database communication model. It consists of an application (client) and a database. [Wutka, 1997]

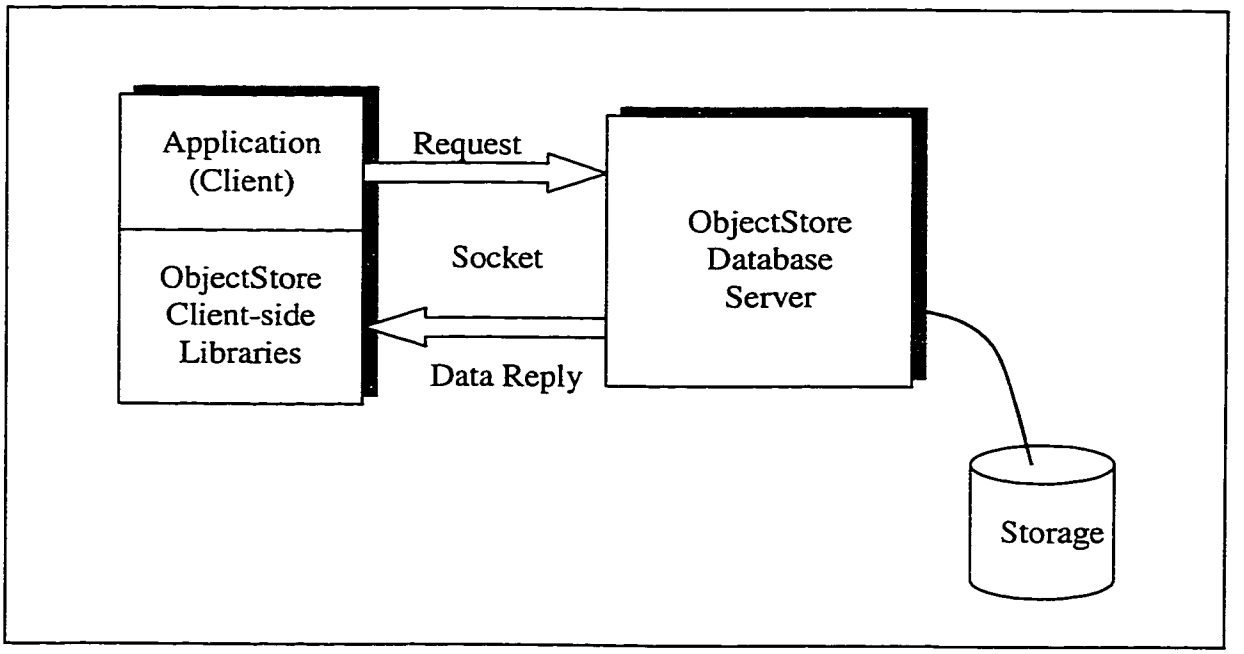


Figure 4.8 The Two-tier Database Communication Model

As shown in the above figure (Figure 4.8), client software is connected to the database through a socket connection. Client program (supplying a user interface) sends request to the Database server. The server returns the appropriate results, and the client is responsible for formatting the data. Clients use ObjectStore client side library of functions that manage the communication between client and server. The advantage of the two-tier model is that application developers do not need to worry about the communication between client and database server. But on the other hand, client-side library will put restrictions on using applications based on two-tier database model [Venhelsuwe, 1997] [Glenn, 1997]:

- Users need to install vender-provided client-side library in order to run the application. Switching from one database vendor to another requires rewriting a significant amount of code for the client application.
- Version control is another issue. When the vendor updates the client-side libraries, the applications that use the database must be recompiled and redistributed.
- User interface and business processing trend to get rolled together, especially with the rapid application development tools on the market. With the user interface so closely tied to business processing, changes to one end up having a direct impact on the other, making maintenance a headache.
- With all this redundant processing occurring on many client machines rather than in a central location, new applications are forced to reinvent the wheel when dealing with the same business processing.

The above restrictions prevent us from using two-tier database communication model to design our TeleLearning system. It is unreasonable to force users to install ObjectStore client-side library before running our system. With the guaranteed execution of the Java Virtual Machine and an easy-to-use Internet socket interface, three-tier database communication model is brought into picture.

4.3.2 Three-Tier Database Communication Model

The three-tier database communication model consists of an application (client, GUI), a tier of business logic (intermediate server), and a database. [Wutka, 1997] [Venhelsuwe, 1997] Figure 4.9 illustrates a three-tier database communication model. The client communicates with an intermediate server that provides a layer of abstraction from the DBMS. The intermediate layer is designed to handle multiple client requests and manage the connection to one or more database servers. This tier also separates business processing from the visual representation of data and knows how to find and manipulate information. The client which evolves into such a learner application is responsible only for retrieving information from the intermediate server and displaying it on the screen. In MICR system, a Java applet was implemented as front-end application. Distance learner shares this applet from his/her web browser to communicates with the intermediate server which is a multi-threaded, standalone, Java application. The intermediate-tier design model provides several advantages over the two-tier design model. The middle tier has the following benefits:[Venhelsuwe, 1997]

- The intermediate server communicates with one or more database server. This allows us to take advantage of distributed database to store our course objects.
- The user of the system does not need to install database client-side library in order to communicate with the database. Version control is not a problem any more.
- It is multithreaded to manage multiple client connections simultaneously. This tier can accept connections from clients on a variety of vender-neutral protocols (from

HTTP to TCP/IP), then marshal the requests to the appropriate vender-specific database servers and return the replies to the appropriate clients.

- The designer can program the middle tier with a set of "business rules" that manage the manipulation of the data. Business rules may include anything such as, user authentication checking, restricting access to certain portions of data, etc.

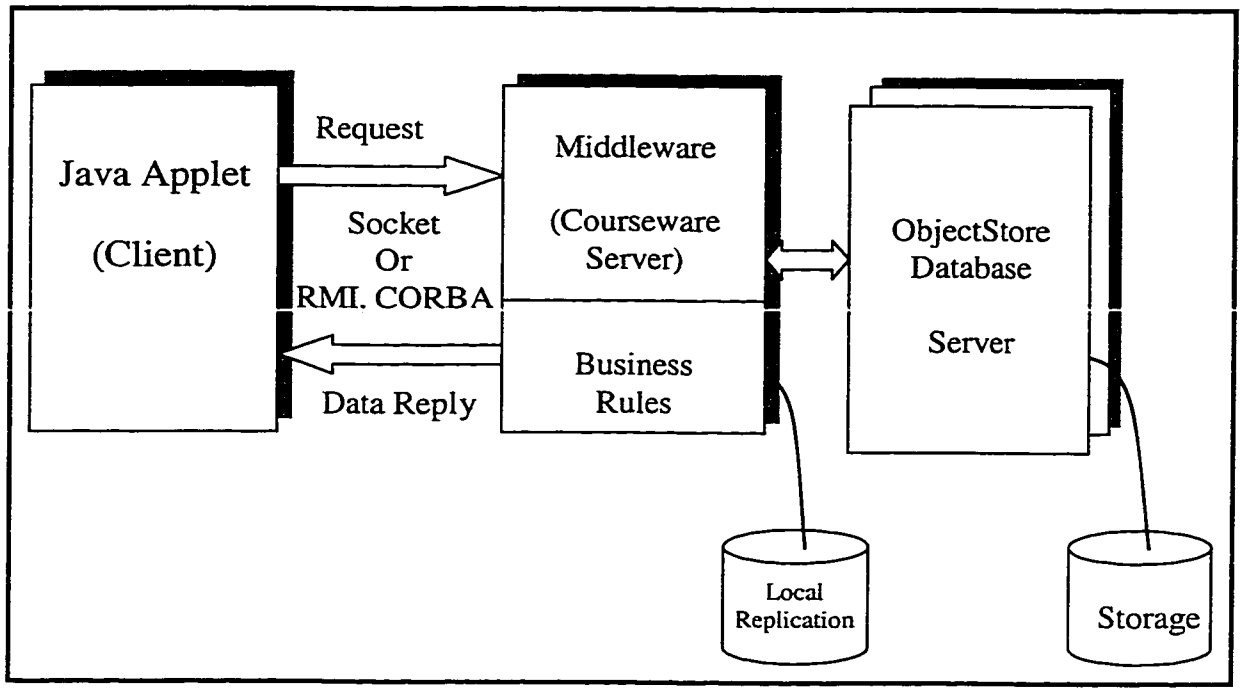


Figure 4.9 Three-tier Database Communication Model

In MICR, communication between clients and the database server was designed and implemented based on the three-tier database communication model. The front-end application is a Java applet that can be downloaded from any point on the Internet with a Java-enabled web browser. Users do not need to install any database client-side libraries in order to communicate with the database. However, with the three-tier architecture,

one of the greatest programming challenges is getting the three layers to communication with one another. ObjectStore Java API or some similar set of database access classes should handle the intermediate server-to-database server communication in a manner transparent to the application developer. The intermediate server translates the requests from clients into specific calls required by the database it supports and calls the specific database driver to handle the requests. Front-end application can be written once and connected to various kinds of database. The application remains the same; the drivers change. This allows the application to get data from back-end database without concern about the database type.

The best two ways for providing client-to- intermediate server communication in Java are *Java sockets* and *distributed objects* (RMI, CORBA). Currently, in MICR system sockets are utilized to provide communication between the applet (Client) and the intermediate server application (Courseware Server). Requests from clients and data reply are transmitted through the socket. Distributed objects such as RMI, CORBA provide the more elegant solution. The developer handles communication simply through methods calls.

4.4 summary

MICR system architecture was designed based on three-tier database communication model and a set of general specifications in this chapter. Compared with two-tier database communication model, the middleware in a three-tier model allows us to take

advantage of distributed database to store our course objects and allows different type of database to be integrated into the system without changing the front-end applications. Another big advantage for three-tier model is that clients don't have to install client-side library in order to communicate with the database.

MICR system architecture consists of a front-end application, namely, the Courseware Controller, back-end database and a middleware, namely, the Courseware Server which is composed of three major components: Client Access Manager, Courseware Rendering Manager, and Java Web Server. The Courseware Controller was designed as a Java applet to help user to access the course materials which stored in our database through the Internet. The Courseware Server keeps running and serving requests from clients and communicating with database. The Client Access Manager is designed as a java application and responsible for establishing connection between clients and the Courseware server. The Courseware Rendering Manager is designed as a Java servlet and responsible for constructing HTML page on the fly according to client's presentation requests.

Chapter 5
System Implementation

5.1 introduction

Using the system architecture presented in chapter 4, a Multimedia Interactive Courseware Rendering System (MICR) was implemented in the University of Ottawa's Multimedia and Mobile Agent Research Lab (MMAR) as part of MITS (Multimedia Interactive TeleLearning System) project. The MICR system successfully demonstrated a number of key concepts described in the thesis, such as platform-independence, interactivity, courseware-on-demand, learner-centered learning environment, etc. It is a successful example in TeleLearning research.

The client applet: Courseware Controller (CWC), and server application: Client Access Manager (CAM), as described in chapter 4, were successfully implemented using Java

API, version 1.0, and ObjectStore 5.0 under Windows NT environment. Although ObjectStore 5.0 also provides C++ API and allows server which accesses database to be implemented in C++, Java was used for the MICR to make the server platform independent as well.

The Courseware Rendering Manager (CRM) described in chapter 4 was implemented as Java servlet using Java Servlet Developing Kit (JSDK) 1.0. It was demonstrated to provide better services which were examined in chapter 3 than CGI. The CRM constructed HTML page on the fly based on the user's content presentation preferences [Falchuk, 1995].

A number of applets ware implemented for rendering multimedia object such as audio and image. They provide users with interactivity at the user-content level [Huang, 1998]. The remainder of this chapter is organized as follows: in the next section, the implementation of client is described, followed by the implementation of courseware server. Some snapshots of the GUI from a learning process are presented in section 4. Finally, the integration with ObjectStore is examined.

5.2 The Client Implementation

5.2.1 Overview

This section describes the implementation of MICR Client. As mentioned before, MICR client consists of two components: Courseware Controller (CWC) which is a Java applet, and a Java-enabled web browser, such as Netscape Navigator from Netscape Communications Corporation, Internet Explorer from Microsoft. The Courseware Controller is embedded in an HTML file managed by the Java Web Server. The HTML file must include the following information:

```
<APPLET CODEBASE = applet  CODE = Controller.class
        WIDTH=450 HEIGHT=150>
<param name=servHost  value="vega.genie.uottawa.ca">
<param name=servPort value="1203">

</APPLET>
```

When the browser encounters the above code in an HTML file, it fetches the applet's corresponding class: `Controller.class` from the code base specified by the `CODEBASE` parameter, and run the applet in a 450X150 pixel^2 area inside the browser. As previously mentioned, when the CWC is launched in the browser, it sends a connection request to the Courseware Server to establish a socket connection. To do this, the applet must be told the host name and the port number which it will connect to. Those values are from HTML's `param` parameters named: `servHost`, `servPort`, respectively. After the applet launched in the browser, the user needs to login the system

from the applet in order to use the CWC to control his/her learning process. User's authentication information is sent to the Courseware Server. Figure 5.1 shows the CWC launched from Netscape Navigator.

In the following section, class hierarchy diagram is presented and explained; then some major classes are discussed.

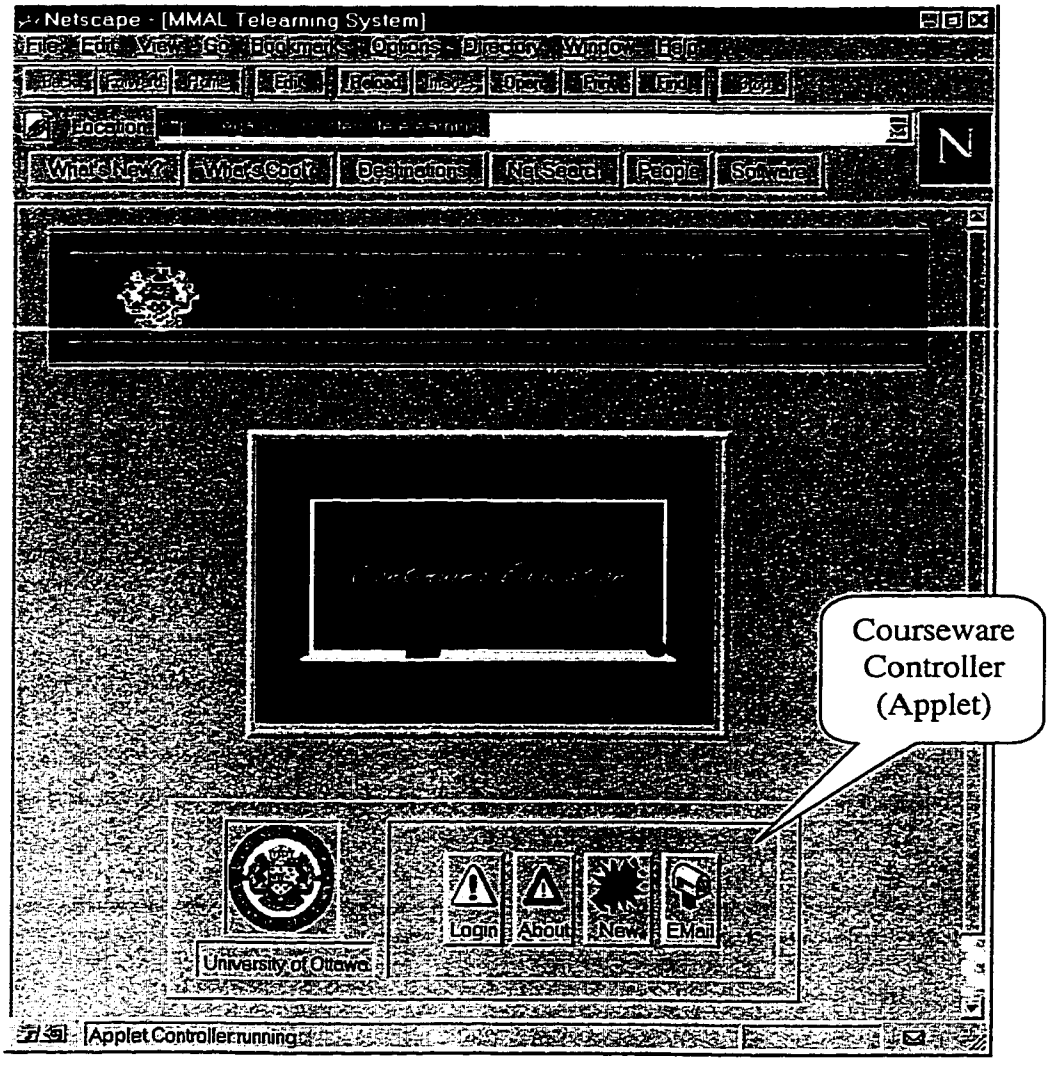


Figure 5.1 Courseware Controller launched in Netscape Navigator

5.2.2 Client Class Diagram

A class diagram is used to show the existence of classes and their relationships in the logical view of a system. In this thesis, all class diagrams are designed based on Booch's design notation [Booch, 1991]. Figure 5.2 illustrates class diagram for Controller related classes

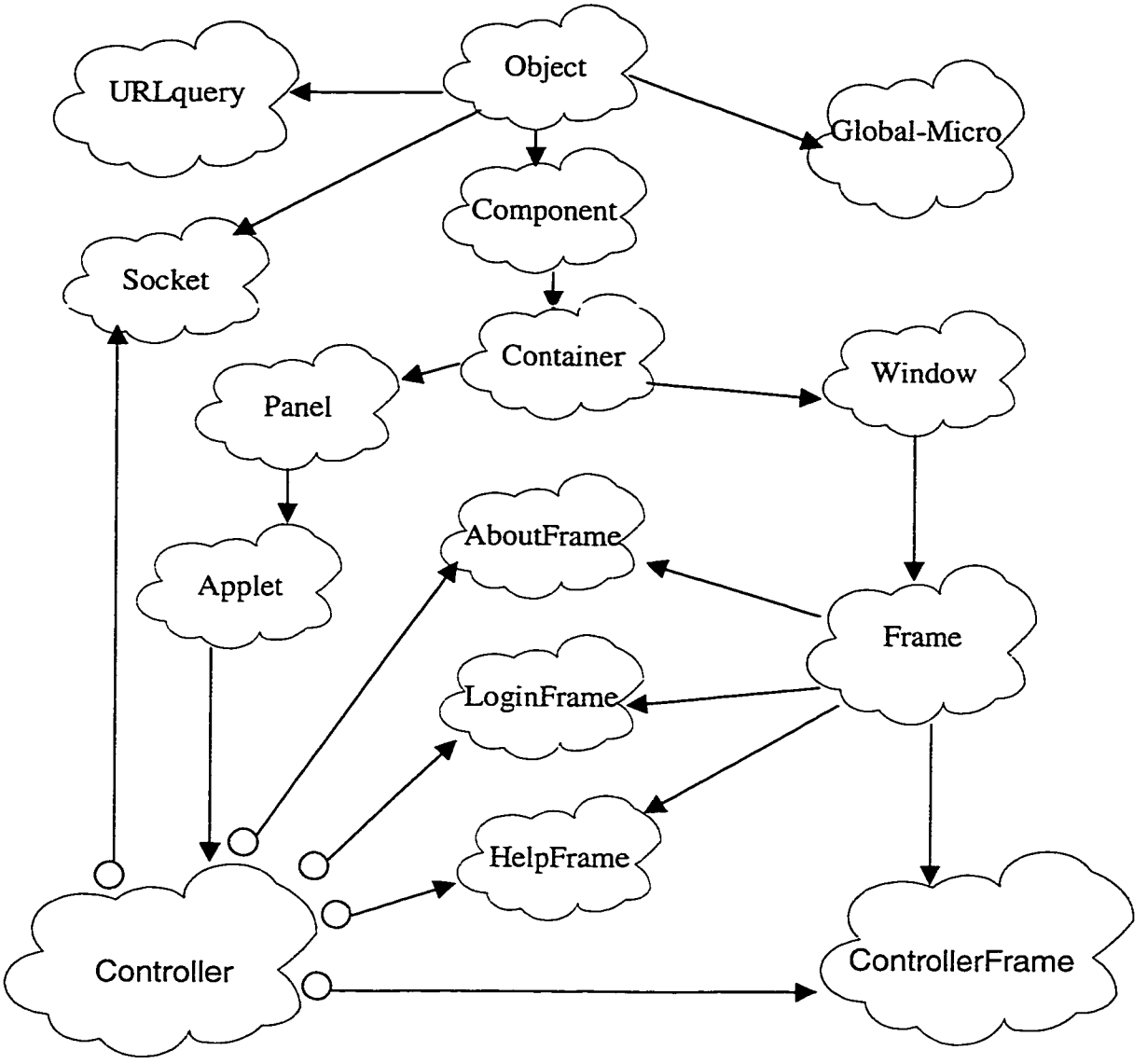


Figure 5.2 Class Hierarchy Diagram for Controller related classes

As illustrated in the above figure, the client is built upon four major classes: **Controller** class, **ControllerFrame** class, **URLquery** class and **Global-Micro** class. They inherit directly or indirectly from base class **Object** which is defined in Java Platform Core API [Sun, 1997]. In Java Core API, class **Object** is the root of the class hierarchy. Every class has **Object** as a superclass.

5.2.2.1 Controller Class

The Controller class inherits `java.applet.Applet` class which inherits from several classes in the AWT (advanced Window Toolkit) package.

```
public class Controller extends Applet
{
     String  servHost;
     int     servPort;
     private Socket sock;
     private DataInputStream datain;
     private DataOutputStream dataout;
     ControllerFrame controllerFrame = null;

     public  void init()
     private void makeConnection()
     public  void MakeRequests(String req_type)
     public  boolean action(Event evt, Object arg)
     public  void createControllerFrame()
     public  void destroyControllerFrame()

     public  void paint(Graphics g)
     public  void stop()
     public  void destroy()
}
```

Figure 5.3 Controller Class Definition

`servHost` is the name of the server on which the Client Access Manager(CAM) is running. `ServPort` is the port number that the Client Access Manager is supposed to listen to. These two parameters are loaded from the HTML document where the applet is embedded. When the server name on which the CAM is running and port number are changed, the client applet does not have to be changed. instead, the parameters in HTML document should be changed. `sock` is the socket that connects the client to the Client Access Manager (CAM). When an applet is instantiated from the Controller class, a socket is open between the applet(client) and the Client Access Manager. `Datain` and `dataout` are data input(server's response) and output (client's request for logical structure) stream through the socket `sock`.

`controllerFrame` is an object instantiated from the `ControllerFrame` class. It is Java frame, illustrated in Figure 5.4, functioning as a control panel for distance learner to control his/her learning process. Details about `ControllerFrame` are discussed later in this chapter.

The Controller class inherits and overrides some essential methods from its parents class for creating applet, such as, `init()`, `stop()`, `destroy()`, `paint()`. Because we did not provide a constructor for the Controller class, all initialization task should be placed in `init()` method. The Controller provides another two important methods for making connection to server and managing the client's request and server's response:

- `makeConnection()`
- `MakeRequests(String req_type)`

During the initialization of the Controller applet, `makeConnection()` method is called to make the connection between the applet and the Client Access Manager by establishing a new socket. After the socket connection is established, `MakeRequests(String req_type)` method is called to send client's request to the Client Access Manager.

5.2.2.2 ControllerFrame Class

`ControllerFrame` class inherits **Frame** class in Java AWT package. It is responsible for constructing a graphical control panel (see Figure 5.4) to help distance learners to control the learning process.

`ControllerFrame` class works with a group of other classes to construct the control panel. The Class hierarchy diagram for `ControllerFrame` related classes is depicted in Figure 5.5. As illustrated in the diagram, `ControllerFrame` class inherits, through several classes in the AWT package, from the root class **Object**. The diagram also indicates that the class `ControllerFrame` is an aggregate, whose instances contain one `StatusBar`, one `MenuBar`, one `MainCardPanel`, and one `Toolbar` which contains all control buttons: `courseButton`, `aboutButton`, `browserButton`, `closeButton`, `audioButton`, `videoButton`, `imageButton`, and `threedButton`. `MainCardPanel` is an aggregate too, whose instances contain exactly one `FirstCard`, and one `SecondCard`. `FirstCard` and `SecondCard` in turn are aggregates. `FirstCard` contains one `ConursePanel`

which may contain any number of `HTMLCourseLabels`, one `MessageCardPanel`, and two `MessageLabels`. `SecondCard` contains one `MessageCanvas`, and any number of `HTMLChapterLabels`.

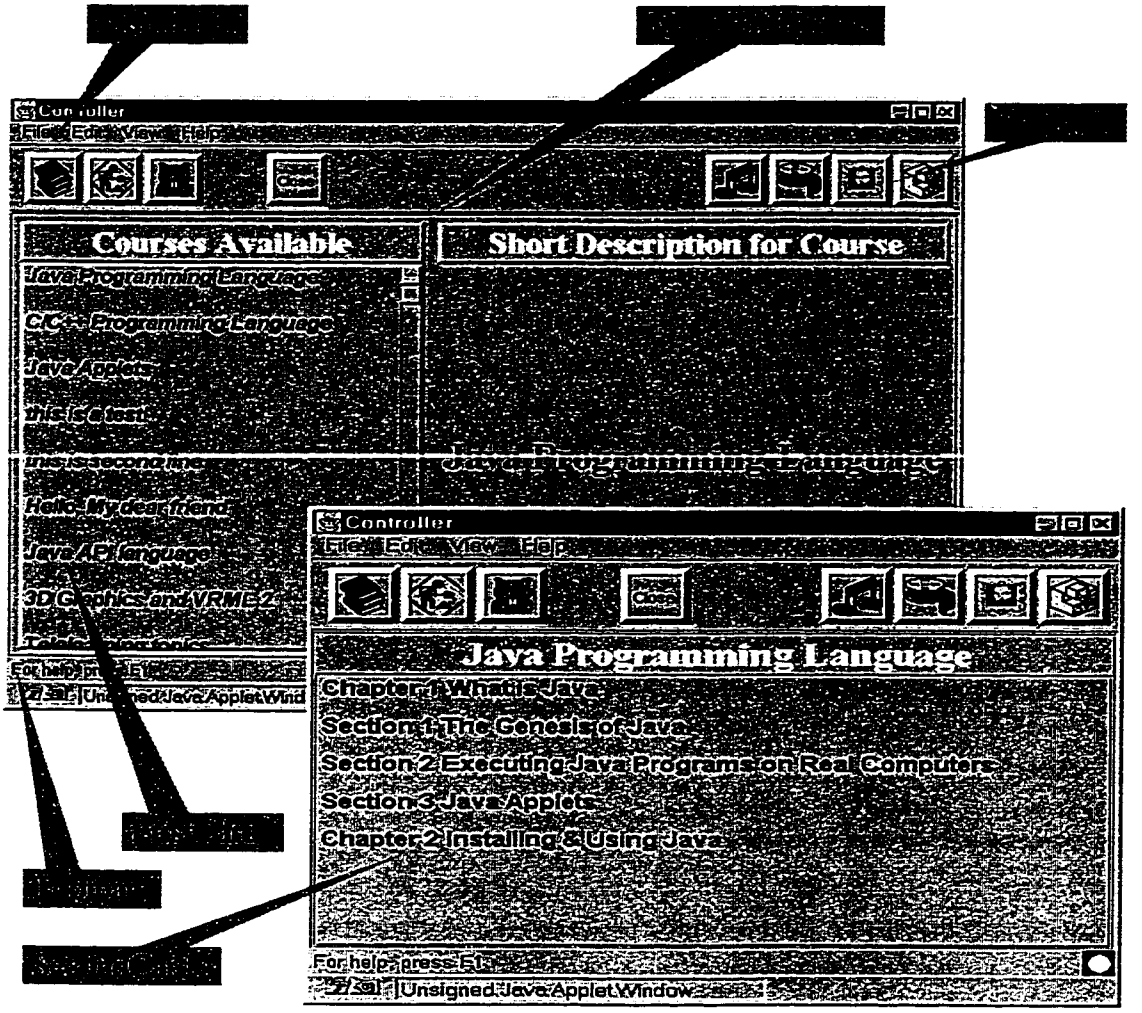


Figure 5.4 Courseware Controller

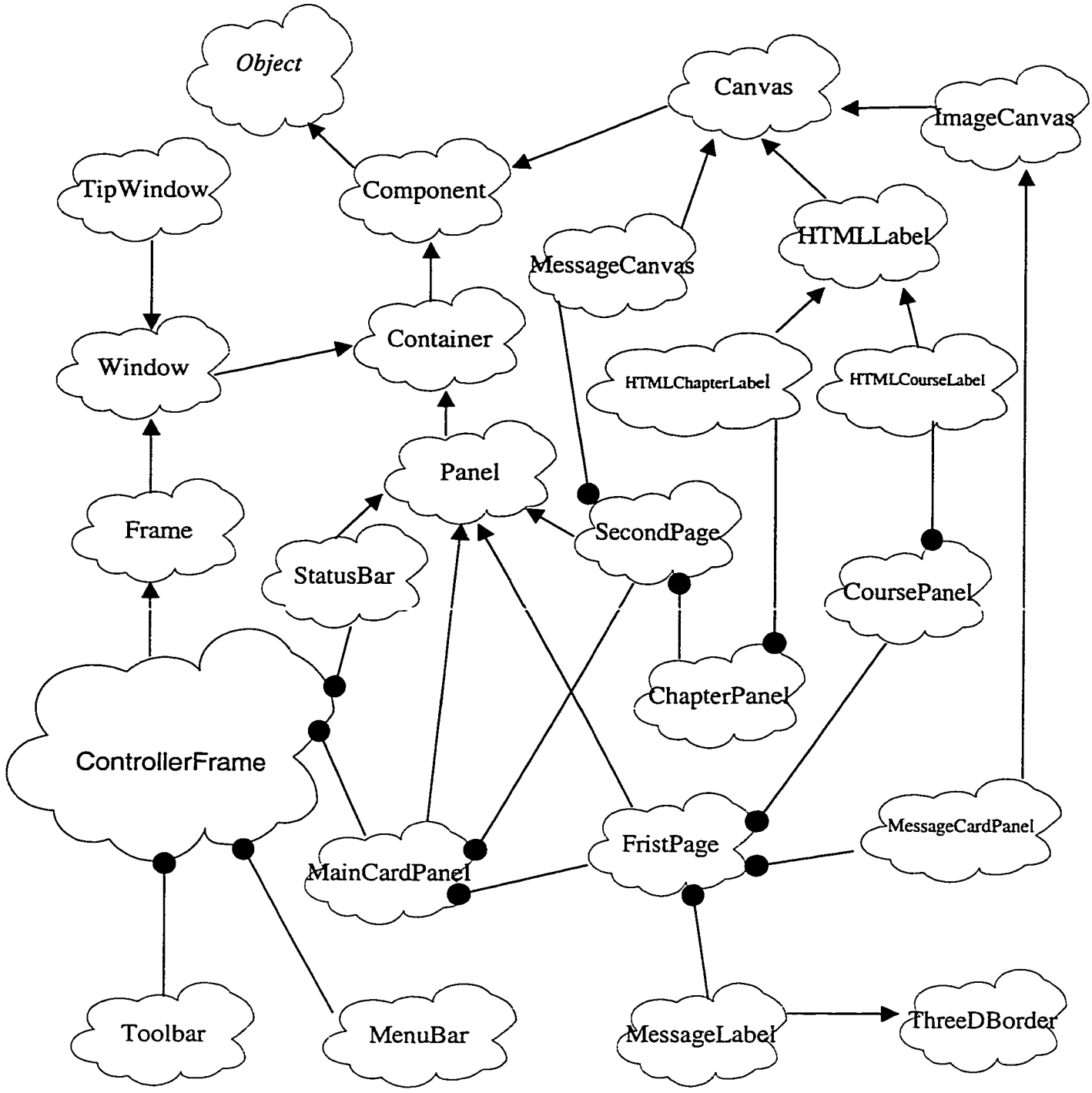


Figure 5.5 Class Hierarchy Diagram for ControllerFrame related classes

`ControllerFrame` class definition is shown in Figure 5.6. The constructor of `ControllerFrame` takes one parameter: applet. This allows the instance of ControllerFrame to communicate with the Controller applet. *applet* is an instance of Controller class. As mentioned previously, when a user launches the Controller applet in his/her browser, the applet makes a connection to the Client Access Manager and gets logical structures from the database and buffers them. When the ControllerFrame popup up, it gets logical structure from the buffers, and displays them in the frame.

```
public class ControllerFrame extends Frame
{
        Controller applet;
        Toolbar toolBar;
        StatusBar statusBar;
        MainCardPanel mainCardPanel;
        TipWindow tip = null;
        Menu fileMenu, editMenu, viewMenu, viewMenu, helpMenu ;
        MenuBar menuBar;
        MenuItem miList, miBrowser, miAbout, miClose;
        CheckboxMenuItem cmiToolbar;
        CheckboxMenuItem cmiStatusbar;
        ImageButton courseButton, aboutButton, browserButton, closeButton;
        ImageButton audioButton, videoButton, imageButton, threedButton;

        //Constructor
        public ControllerFrame(Controller applet)
        {
                super("Controller");
                this.applet = applet;
        }

        private Toolbar makeToolbar()
        public MenuBar ControllerMenuBar()
        public boolean handleEvent(Event ev)
}
```

Figure 5.6 ControllerFrame class definition

5.2.2.3 URLquery class

```
class URLquery extends Object
{
        public static URL createQuery(URL urlOriginal, Properties propsParam)
                throws MalformedURLException
        {
                String protocol = urlOriginal.getProtocol();
                String host = urlOriginal.getHost();
                int port = urlOriginal.getPort();
                String file = URLEncoder.encode(strPropsName) ;
                return new URL(protocol, host, port, file);
        }
}
```

Figure 5.7 URLquery Class Definition

`URLquery` class creates an URL to perform a query against a Web server. When a user selects a course to read from the Courseware Controller, the `URLquery` class is called to create URL query. The query will be sent to the Java Web Server on which the Courseware Rendering Manager resides. There is no constructor for this class. Only one public method is provided in this class: `createQuery(URL urlOriginal, Properties propsParam)`. This method takes two parameters: `urlOriginal, propsParam`. `urlOriginal` is a *URL* type parameter which indicates where the Java web Server is running. `propsParam` is a *Properties* type parameter that contains a set of properties that will be converted into a query string. The set of properties is from the user's presentation preference, such as: which course, which chapter of the course,

which section of the chapter the user is selected? What kind of media object will be included in the presentation: video, audio, image, or VRML virtual world? The result returned from the method is a URL object from the specified protocol, host, port number, and file.

URLEncoder is a system class from java.net package. The class contains a utility method: `encode(String str)` for converting a String into a MIME format called "x-www-form-urlencoded" format.

5.2.2.4 Global-Micro Class

```
class Global_Micro extends Object
{
        public final static String URL_COURSE_FINDER =
                "http://vega.genie.uottawa.ca:8080/servlet/FrameServlet";
}
```

Figure 5.8 Global_Micro Class Definition

Global_Micro is a simple class functioning as a global class. It defines a public, final, static, string: URL_COURSE_FINDER indicates where the Courseware Rendering Manager is. Once the server host running the Courseware Rendering Manager is changed, we only need to modify the Global_Micro class instead of other classes which using the server information.

5.3 Implementation of the Courseware Server

5.3.1 Overview

As stated in Chapter 4, the Courseware Server consists of three components: Client Access Manager, Courseware Rendering Manager, and Java Web Server. They work together to serve distance learners both logical structures of courses in database and course contents requested by users. Based on the server design described in chapter 4, the implementation of components that comprise the Courseware Server is presented in this section. In addition to describing classes diagrams based on Booth's design notation, details are provided of definitions of some important classes comprise the Courseware Server, such as `Server` class, `ServerThread` class, and `FrameServlet` class.

5.3.2 Class Diagram for Client Access Manager

Figure 5.9 depicts the class hierarchy diagram for the Client Access Manager related classes. As we can see from Figure 5.9, the `Server` class aggregates two classes: `Socket`, and `ServerSocket` which are from Java core API. The `Socket` class implements client sockets. A socket is an endpoint for communication between two machines. The `ServerSocket` class implements server sockets. A server socket waits

for requests to come in over the network. It creates a new thread based on that request, and then returns logical structures to the requester. Both `Socket` class and `ServerSocket` class extends Java root class **Object**. `ServerThread` class extends class `Thread`. Each ServerThread contains exactly one `Socket`.

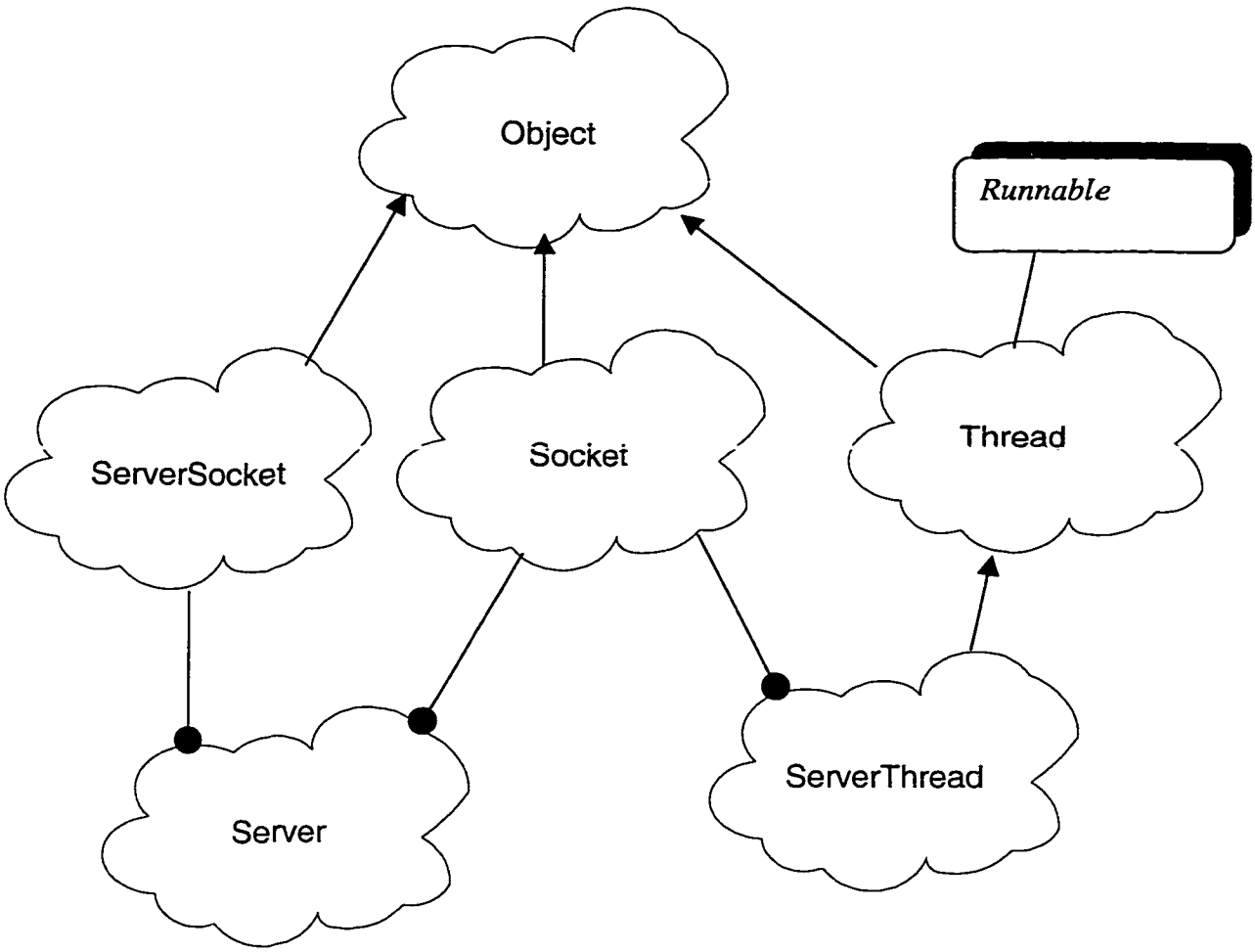


Figure 5.9 Client Access Manager Class Hierarchy Diagram

5.3.2.1 The Server Class Definition

```
public class Server
{

        static ServerSocket ServSock;
        static Socket theSocket;

        static ServerThread client[] = new ServerThread[NumberofClients];

        DataInputStream datain;
        DataOutputStream dataout;
        static int port = 1203;


        public static void main(String args[])

        static void ReadTeleDB(Database db)

}
```

Figure 5.10 Server Class Definition

ServSock is a `ServerSocket` type socket. It listens requests from clients and opens one socket *theSocket*, and one thread *ServerThread* for each accepted client. *client[]* is an array of ServerThread objects corresponding to the number of client socket. *NumberofClients* indicates the maximum number of clients allowed to access the server at simultaneously. This is used to prevent overwhelming of the server. This number depends on the resources of the CPU on which the server runs. *port* is the port number

that the server is supposed to listen to. *datain* and *dataout* are the input and output of the socket (data channel).

The Server class (Client Access Manager) is a standalone Java application. It must provides a *main()* method. The *main()* method performs the following actions:

- ObjectStore initialization
- Opens database (ObjectStore)
- Call method *ReadTeleDB()* to start a read only transaction to retrieve database (ObjectStore)
- Cache logical structure retrieved from database (ObjectStore)
- Close database
- Accept or reject client connection request
- Open a socket and starts a client thread for each accepted client

ReadTeleDB() is another important method provided in the `Server` class. This method must contain code for operating database, such as starting a transaction, retrieving data from database, closing a transaction, etc..

5.3.2.2 The ServerThread Class Definition

The definition of class `ServerThread` is described in Figure 5.11.

Once the `Server` class accepted a client's connection request, the ServerThread class is called to instantiate a thread object for this client. This thread processes and responses the client's request.

```
public class ServerThread extends Thread
{
        private Socket mySocket;
        private DataInputStream datain;
        private DataOutputStream dataout;
        private int myId;


        // constructor
        public ServerThread(Socket mySocket, int myID, Vector courses ) throws
                        IOException

        public void run()

        private boolean processRequests() throws IOException

}
```

Figure 5.11 ServerThread Class Definition

mySocket is the socket for this client. All input data stream *datain* and output data stream *dataout* for this client are transmitted through this socket. MyID is an integer that is used by the server to decide whether accepts this client or not. The constructor for the `ServerThread` class takes three parameters: `Socket` *mySocket*, `int` *myID*, `Vector` *courses*. As mentioned previously, the `Server` class uses its main thread to initialize and retrieve ObjectStore and caches the results retrieved from the database in order for each client to save time. Clients can get logical structure from cache instead of from database. `Vector` *courses* is the cache. Also, stated in Chapter 4, using the main thread in the Server class to retrieve database and cache the result is superior, when the database is not big. But when the database becomes bigger, this way is not practical. So, the

second way comes into picture: allow each client thread to initialize database and retrieve information requested by this client. This part of code should be included in the method *run()*.

5.3.3 Class Diagram for Courseware RenderingManager

Class `FrameServlet` is a Java Servlet functioning as Courseware Rendering Manager. A servlet is a body of Java code that is loaded into and runs inside a network service, such as a web server. (In our case, we use Java Web Server from Sun Microsystems) [Servlet, 1997] It receives and responds to requests from clients(Courseware Controller). For example, a client may need information about specific course from a database; a servlet can be written that receives the request, gets and processes the data as needed by the client, and then returns it to the client.

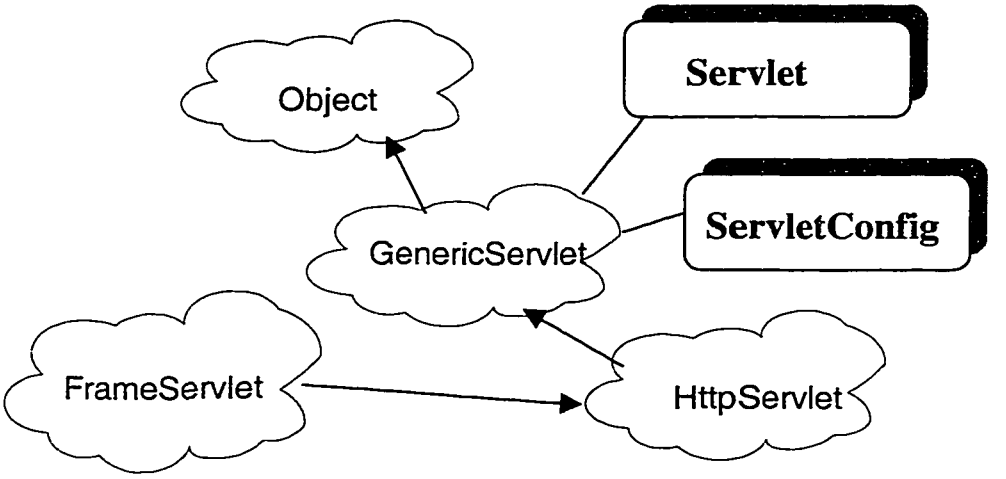


Figure 5.12 Class Diagram for CRM

As illustrated in Figure 5.12, `FrameServlet` inherits class HttpServlet which inherits class `GenericServlet`. The `GenericServlet` class implements the Servlet interface and, for convenience, the `ServletConfig` interface. The Servlet interface defines methods to initialize a servlet, to receive and respond to client requests, and to destroy a servlet and its resources. `ServletConfig` interface is implemented by services in order to pass configuration information to a servlet when it is first loaded. The GenericServlet class was created to make writing servlets easier. It provides simple versions of the lifecycle methods `init()` and `destroy()`, and of the methods in the `ServletConfig` interface. The `HttpServlet` class is an abstract class that simplifies writing HTTP 1.0 servlets. It extends the `GenericServlet` base class and provides a protocol handling framework.

The functions of `FrameServlet` class can be described as the following four modules, illustrated in Figure 5.13.

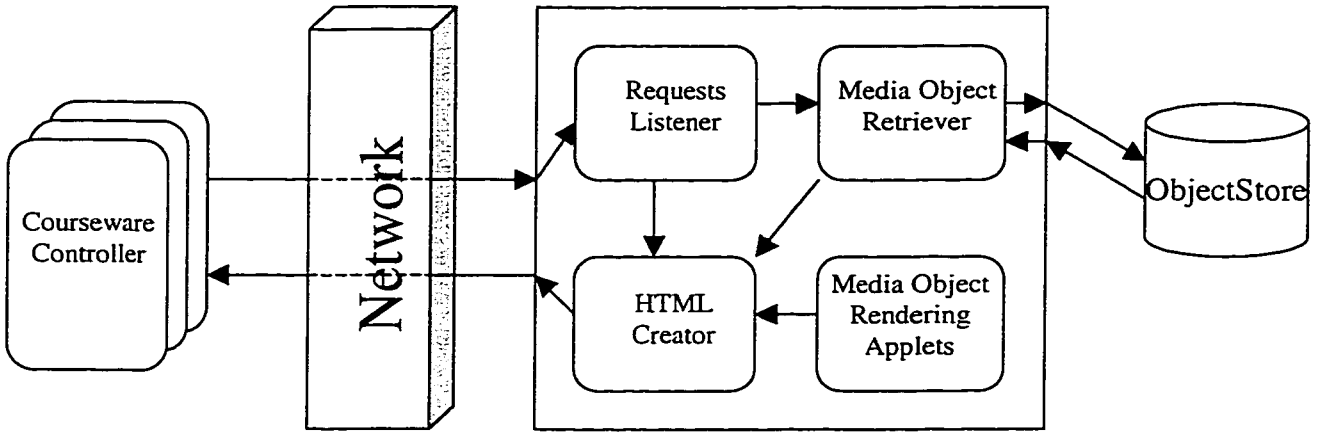


Figure 5.13 FrameServlet Function Diagram

Request Listener contains codes to parse requests coming from clients over network. Media object Retriever uses ObjectStore Java API to retrieve ObjectStore database based on clients' requests. Media Object Rendering Applets try to provide client with some kind of interaction, such as, drag and drop, click and double click mouse, during the process of media object rendering. HTML Creator contains code to create HTML document on the fly based on clients' presentation preference.

FrameServlet class definition is illustrated in the following:

```
public class FrameServlet extends HttpServlet
{
        public void init() throws ServletException

        public void service(HttpServletRequest req, HttpServletResponse resp)

        private void listener()
        private void optionCodeParser()
        private void ReadDB()
        private void HTMLCreator()
}
```

5.4 System Integration

As described previously, all courseware objects are stored in an object-oriented database, namely, ObjectStore, which is a commercial product from Object Design. The front-end application, Courseware Controller, gets information from database through middleware applications, the Client Access Manager and the Courseware Rendering Manager. After

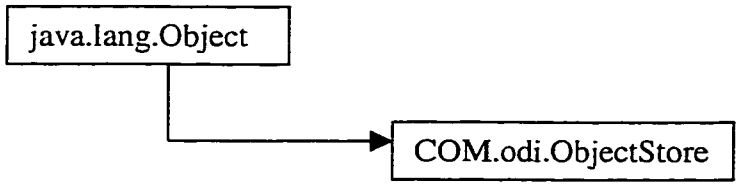
the discussion of system design and the implementation of the front-end application and middleware application, it is time to integrate our database into the system.

5.4.1 Some Basic ObjectStore API for Integration

ObjectStore provides Java API for Java applications to operate database smoothly. The integration to ObjectStore means integrating the API provided by ObjectStore for operating database into middleware applications, i.e. the Client Access Manager, and the Courseware Rendering Manager. The following describes some important classes that are used by the Client Access Manager and the Courseware Rendering Manager.

- **Class ObjectStore**

Class `ObjectStore` is from Package COM.odi. The ancestry of class ObjectStore is as follows:

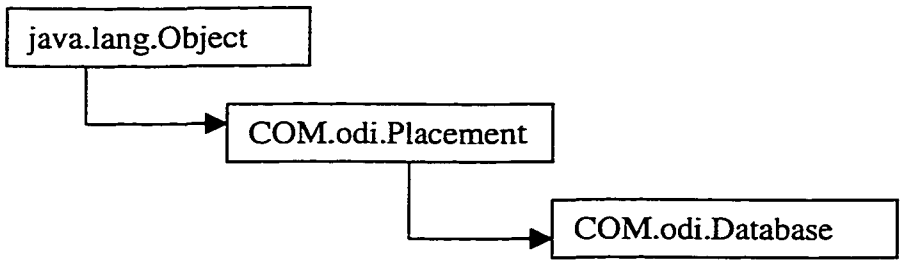


The `ObjectStore` class defines system-level operations that are not specific to any database. When you start a session to operate ObjectStore, the method `initialize(String, Properties)` or `initialize(Thread)` of the class

must be called. Both of them initializes a ObjectStore session, which permits the caller to use the rest of the API, such as `Database`, `Transaction`, and so on.

❖ Class Database

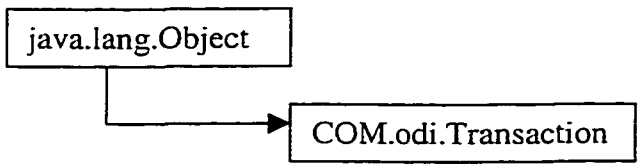
The ancestry of class `Database` is as follows:



Placement is a abstract superclass of classes that specify where an object should be migrated or allocated. The `Database` class, which inherets this class, is an abstract class that represents a database. For each database you create or open, ObjectStore automatically finds the subclass of Database and creates an instance of that subclass. ObjectStore then associates the instance with your database. The instance represents your database and provides a handle to your database. This class provides a rich set of variables and methods for developers to make business rules for operating database, such as, read database permission, write database permission, and so on.

❖ Class Transaction

The ancestry of class `Database` is as follows:



ObjectStore uses the Transaction class to represent a logical unit of work. A transaction is a consistent and reliable portion of the execution of a program. In your code, you place calls to the ObjectStore API to mark the beginning and end of transactions. In an application, the initial access to a persistent object must always occur inside a transaction. For more detail information about those class, you can reference to ObjectStore Java API Reference [ObjectStore, 1997].

5.4.2 Persistence -Aware process

As you probably know, Persistence-capable is the capacity to be stored in a database. If you can store an object in a database, the object is persistence-capable. If you can store the instances of a class in a database, the class is a persistence-capable class and the instances are persistence-capable objects. All the Telelearning courseware objects stored in ObjectStore are persistence capable objects. According to ObjectStore's theory, if the methods of a class can operate on persistent objects but instances of the class itself are not persistence-capable, the class is persistence-aware. Typically, if you want a class to be persistence-aware, you run the postprocessor on it to put in the required annotations.

The Client Access Manager and the Courseware Rendering Manager will operate on persistent objects stored in ObjectStore, but they never need to be stored persistently. They must be persistence-aware. ObjectStore provides a Class file Postprocessor to help to annotate classes and make them persistence-capable or persistence-aware.

5.5 Mode of Operation

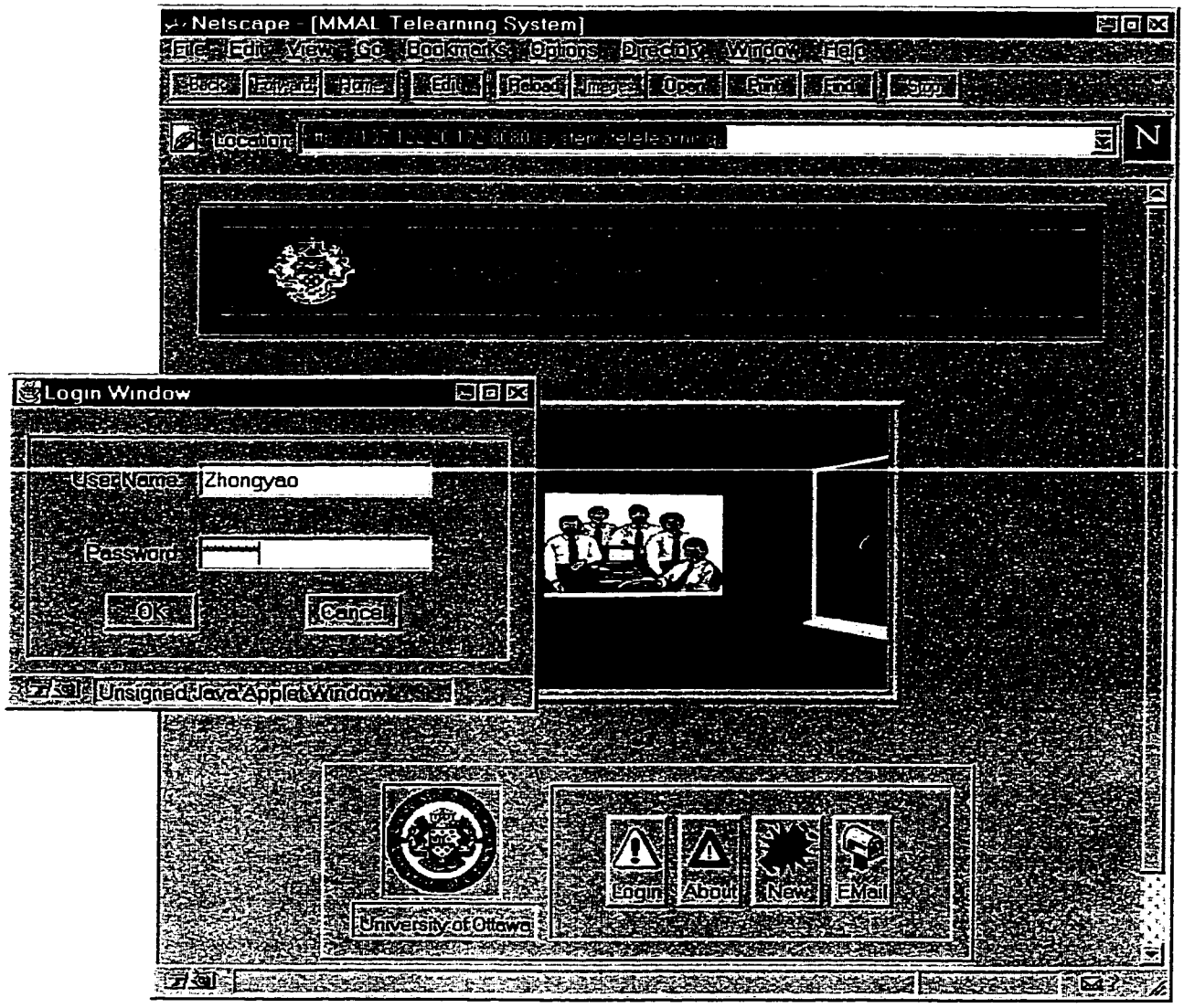


Figure 5.14 MICR Login Page

The first prototype of our Multimedia Interactive Courseware Rendering System for TeleLearning (MICR) was implemented using Java JDK 1.0.2 under Windows NT

environment. Distance learners can access our MICR system from any point on the Internet. But before the user can login the system to browse courses available in our Multimedia Courseware Database, he/she needs an account for accessing the system. Once registered, he/she can use the user name and password to login the system from the following web page, as shown in Figure 5.14.

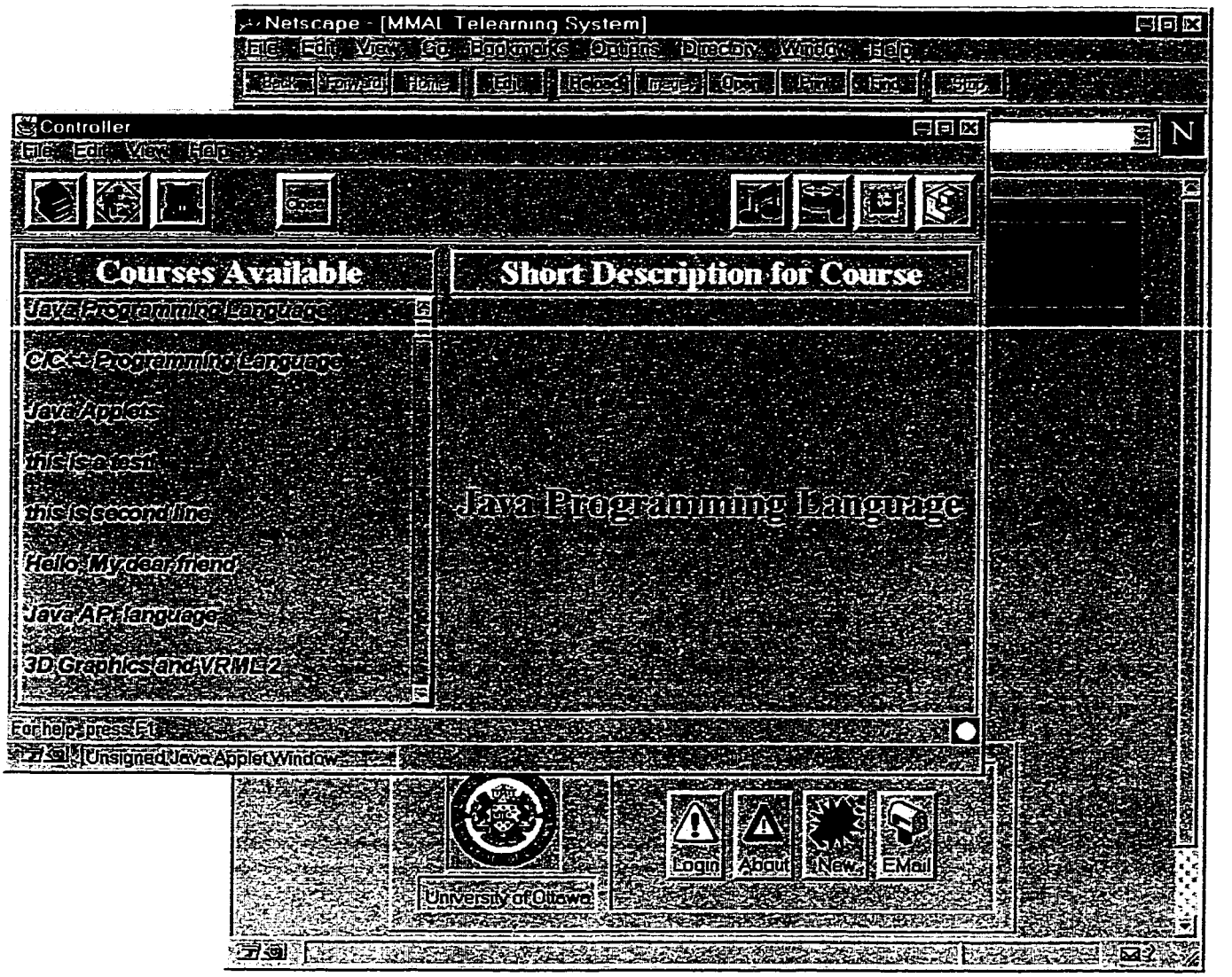


Figure 5.15 Courseware Controller Panel and the Web Browser

After the user click on the login button, the system will ask the user to authenticate himself/herself by poping up a login window (see Figure 5.14) for the user to input user name and password. Once the user authenticates himself/herself, the system will start the Courseware Control Panel for the user, as shown in Figure 5.15. At this time, the Controller connected the user to the Courseware Server, and got the list of all available courses in the Courseware Database. The list is displayed in the control panel, as shown in Figure 5.15.

As we can see, the Controller stands besides the web browser. The controller communicates with the web browser for sending user's requests to the Server. The user can selects any course from the control panel to read. As you can see, there is a short description area in the controller window. (see Figure 5.15) When the user point his/her mouse on the title of a course, a short description for this course is displayed in the short description area to help the user to decide whether this course is what he/she wants. When the user click this course, the logical structure of this course is shown in the controller panel, as illustrated in Figure 5.16.

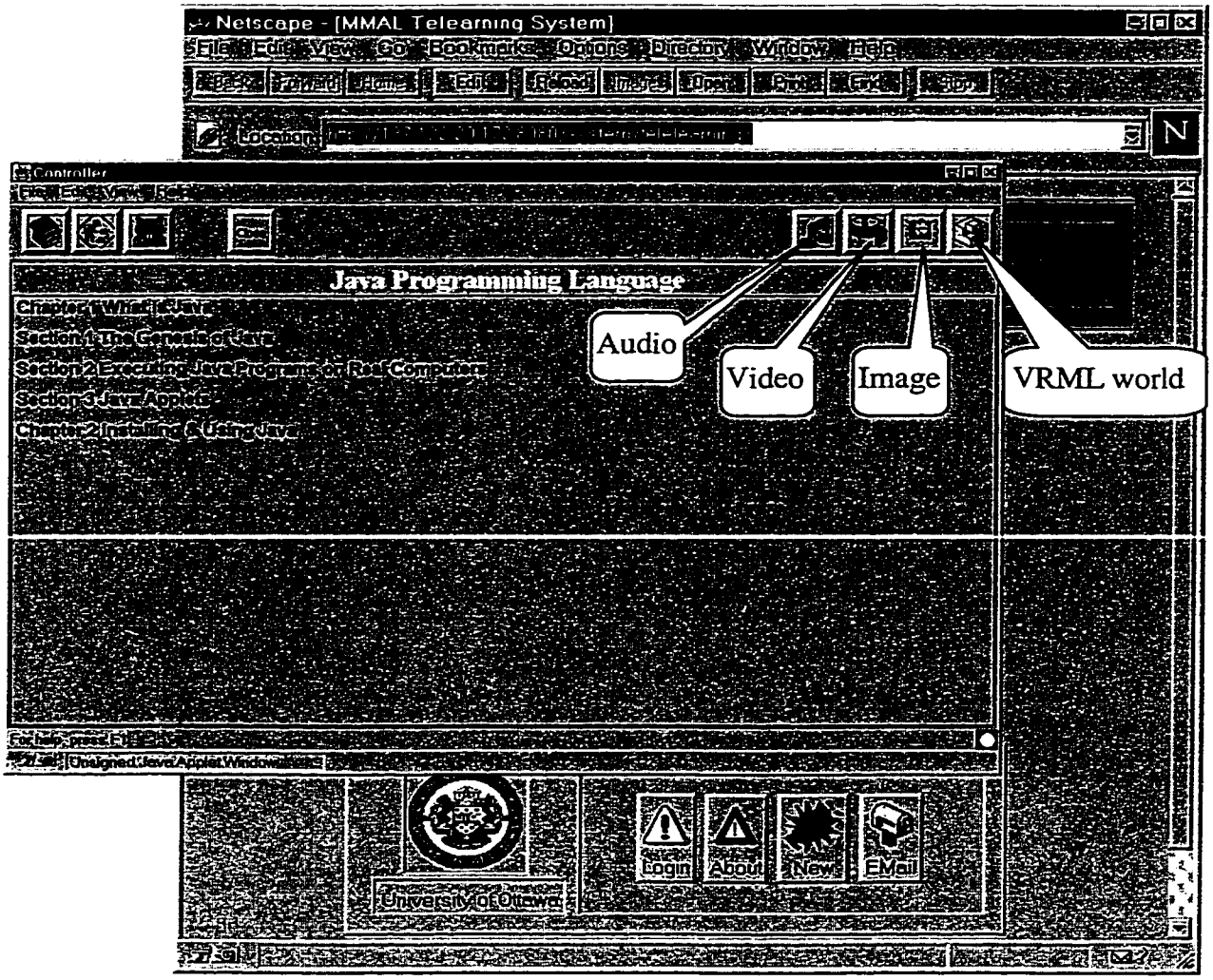


Figure 5.16 Logical Structure Displayed in Controller

You may noticed that there are four toggle buttons in the upper right corner of the Controller which stand for Audio, Video, Image and VRML virtul world media type, respectively. When a button is pressed down, the associated media object will be included in the presentation. For example, the user presses down the Audio button, and

then selects to read section 1 in chapter 1, the content of this section is displayed in the browser, audio objects in this section are transported to the user, as shown in Figure 5.17. The user didn't press down video button, image button, and VRML world button, those media objects will not be transported to the user. You can click play button to play the audio clip, stop button to stop. Figure 5.18 shows another sample page with Video object included. This a MPEG video clip which is played back with the plugin helper application. When the user click his/her mouse on the video window, the plugin will start to play the video clip. During the playback, mouse clicking on the video window will stop and resume the playback.

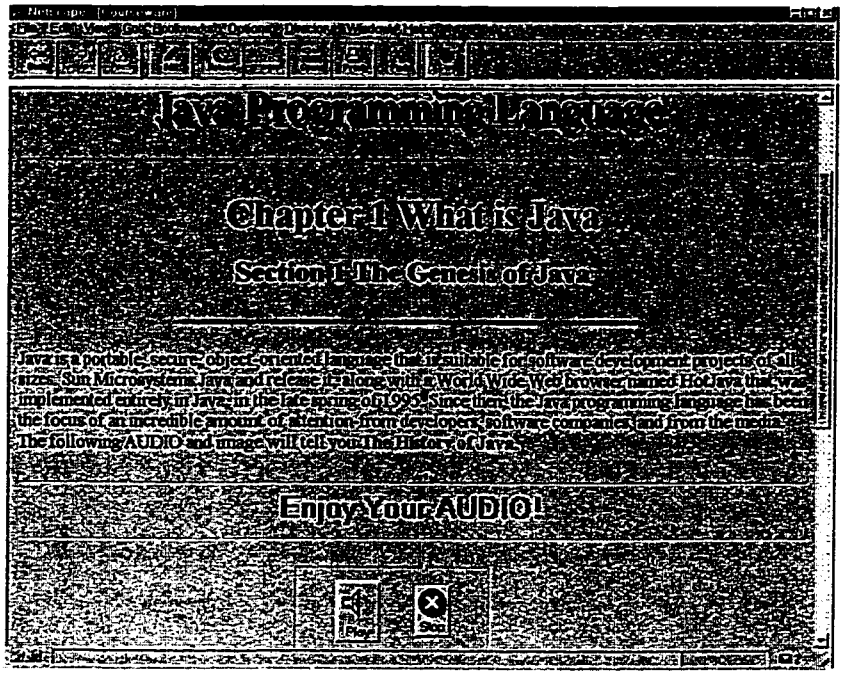


Figure 5.17 A sample page with Audio object included

Figure 5.19 with VRML virtul world included. The VRML plugin will help to playback the VRML world included in the content. Each vritul component inside the virtual world

can be defined as a link connected to a specific location. Click on the component will lead the user to the location defined in the link.

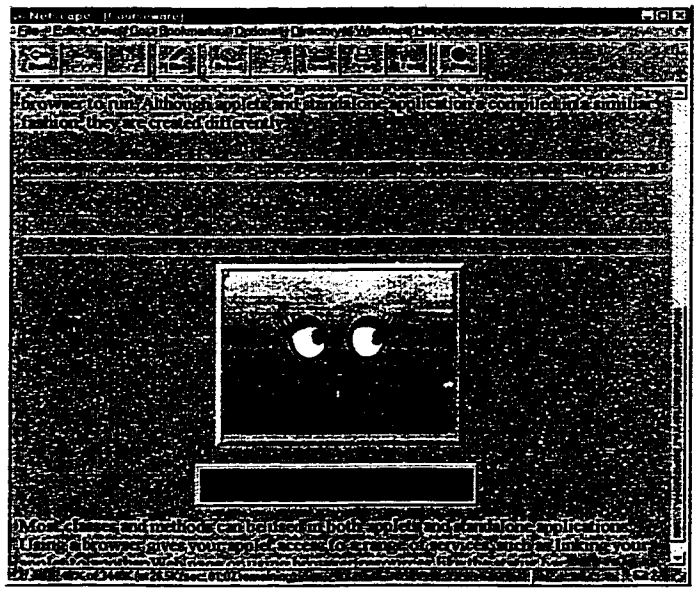


Figure 5.18 A sample page with Video object included

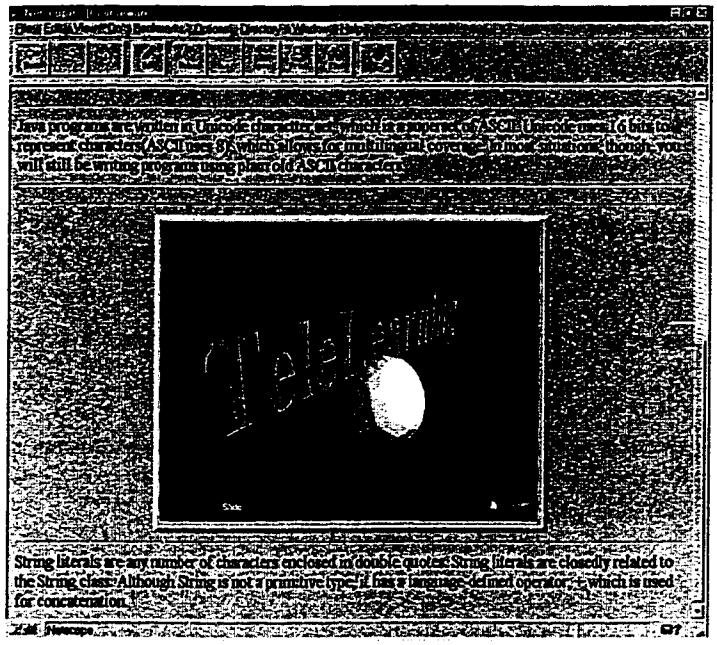


Figure 5.19 A Sample page with VRML world included

5.6 Summary

In this chapter, the implementation of MICR system and the integration with ObjectStore database were presented. The prototype successfully demonstrated a number of key concepts : platform indepence, course-on-demand and learner-centered learning environment.

The Courseware Controller and the Courseware Server, as described in chapter 4, were successfully implemented using java programming language. Once a client download the Courseware Controller which is a Java applet, a connection between the client and the Courseware Server is established. Logical structures for courses are displayed in the Couseware Controller window. Contents of selected course is displayed in browser as HTML document which is constructed on the fly by the Rendering Manager.

Detailed class diagrams for both client side application and server side application were described using Booch's OO design notation. Defination for some major classes were also presented.

The prototype developed at the Multimedia & Mobile Agent Research Laboratory in University of Ottawa successfully demonstrated that the system works as designed. It also showed that the theory and architecture behind the system is valid and implementable.

Chapter 6
Conclusion and Future Work

6.1 Summary

In this thesis, we presented the design, implementation of a platform-independent Multimedia Interactive Courseware Rendering (MICR) System for TeleLearning. The aim of this system was to provide a seamless TeleLearning environment that is accessible by distance learners from any point on the Internet with minimum amount of user-software. This was successfully demonstrated through the implementation of a prototype system. The main features of MICR can be summarized as: platform-independence, course-on-demand, user interactivity, seamless learner-centered learning environment.

The system design of MICR was based on client-server and object-oriented design principle. The system architecture of MICR can be described as three-tier database

communication model. The client communicates with intermediate server that provides a layer of abstraction from the database. This layer isolates the client application from database system and allows different type of database to be integrated in the system without changing the front-end application.

A prototype of MICR has been successfully implemented in our laboratory using Java JDK1.02 (Java Development Kit) and JSDK 1.0 (Java Servlet Development Kit) The Courseware Controller was implemented as a Java applet which ensures that any Internet user equipped with a Java-enabled Web browser has access to the system. This gave our system a valuable feature: platform independence. The Courseware Controller allows the users to select courses they want from our database, and even allow them to decide what kind of media object, such as image, audio, video, and VRML virtual world will be included in presentation based on the users' hardware ability.

The Courseware Server consists of three functional components, namely, Client Access Manager which was implemented as multithread, persistence-aware, Standalone, Java application, Courseware Rendering Manager which was implemented as a Java servlet using JSDK 1.0 to create HTML document on the fly according to the user's presentation preference. The third component of the Courseware Server is Java Web Server which is a commercial product from Microsystem to provide a well-defined environment for the Courseware Rendering Manager. The interactivity provided in this system such as click, double-click, and drag-and-drop was achieved by implementing a group of Java applets

which will be called by the Courseware Rendering Manager during constructing HTML documents.

The database used in our system to store created courses is a commercial product from Object Design: ObjectStore 5.0. It provides Java API for operating database. The integration with ObjectStore was successfully achieved.

6.2 Suggestions for Future Work

While the main objective of this research, design and implementation of a Multimedia Interactive Courseware Rendering (MICR) System for TeleLearning, has been achieved, and a prototype of MICR is in place, there are several improvements that can be made to further increase the quality of MICR system.

- Interactivity

The aim of MICR is to provide a seamless TeleLearning environment for distance learners. Interactivity or communication between teacher and students, between students and course materials, and among students themselves can make students feel like in a "real" classroom and not isolated. In our prototype, we provided interactivity between students and course materials. Future research is recommended to add interactivity between student and online facilitator, among students themselves.

- Real- time

The way to deal with contiguous media object, such as audio and video can be improved. In our prototype, audio objects are handled by classes provided in Java API core package. Video objects are played back using plugins. Both audio and video objects are downloaded first and then play. Real time protocol such as RTP (Real time Transport Protocol), RTCP (Real time Transport Control Protocol) could be used to handle audio and video media type to enable real time communication and collaboration over the Internet. Microsoft NetMeeting™ is a good example [NetMeeting, 1998].

- Improvement of Server Performance

In our prototype, the Courseware Server was implemented using classes provided in Java JDK1.0.2 package. As we probably know, Java is a interpreted language. it provides a lot of good features such as platform independence, security, and so on, but it is slower than C++ in nature, if it is not running on JavaOS, which is an operating system devoted to run Java bytecodes. C++ can be used to implement more efficient server. note that the implementation of the server in C++ does not take away the platform independence feature of MICR client application, the Courseware Controller.

- CORBA or RMI

CORBA (Common Object Request Broker Architecture) and RMI (Remote Method Invocation) are two touchy terms in distributed computing. CORBA and RMI could be introduced in our system to deal with the communication between clients and remote courseware objects stored in our database, ObjectStore. Using CORBA or RMI, client object can invoke methods of remote objects just like those objects are local. RMI share

the same concept with CORBA. RMI is called "pure Java" CORBA. Our prototype was implemented using pure Java, so RMI would be superior to our prototype. If the server is changed to a C++ version in future for better performance, CORBA will be the best choice for our prototype.

- Media Managers

ObjectStore provides Multi-Media Managers, namely, Audio Object Manager, Image Object Manager, Video Object Manager, Text Object Manager, PDF Object Manager, and Java Object Manager. These managers allow application developers to integrate multimedia objects such as, audio, image, video, and text objects into your applications very quickly, without writing code to handle media objects in your application. Media managers will provide efficient way to manage multimedia object used in courseware objects.

References

[Sherry, 1996] L. Sherry (1996) Issues in Distance Learning. International Journal of Distance Education, 1 (4), 337-365

[Huang, 1998] Beilei Huang Enhanced Interactivity in A Multimedia System Over Internet Proceedings of IEEE Canadian Conference. on Electrical and Computer Engineering '98, Waterloo, May 24-28, p.533-536.

[Gu, 1995] Grace Gu, Can Multimedia Help People Learn Faster? Proceedings of the IEEE international conference on multimedia computing and systems, May 1995

[Falchuk, 1995] Ben Falchuk, A. Karmouch, Amultimedia News Delievery System Over an ATM Network, Proceedings of the IEEE international conference on multimedia computing and systems, May 1995

[Maly, 1996] K. Maly, H. Abdel-Wahab, C.M. Overstreet, C. Wild, A. Gupta, A. Youssef, E. Stoica, E. Al-Shaer and R. Talla, Distance Learning and Training over Intranets http://www.cs.odu.edu/~tele/iri/papers/alexardria.ps Department of Computer Science Old Dominion University Norfolk, VA 23529-0612

[Marchal, 1997] Benoit Marchal, Dynamic WebPages in Java –Servlet- http://www.javacats.com/US/articles/servlet.html

[Wutka, 1997] Mark Wutka, David Baker et al Hacking Java: the Java Professionals Resource Kit ISBN 0-7897-0935-x Que Corporation, 201 W. 103rd Street, Indianapolis, IN 46290

[JDK, 1997] Java Development Kit 1.0.2 API, Javasoft, http://java.sun.com/products/JDK/CurrentRelease/api/

[Glenn, 1997] Glenn L. Vanderburg, et al. Tricks of the Java Programming Gurus ISBN 1-57521-102-5 Sams.net Publishing, 201 W. 103rd St., Indianapolis, IN 46290

[Orfali, 1996] Robert Orfali, Dan Harkey et al. The Essential Client/Server Survival Guide. Second Edition ISBN 0-471-15325-7

[Roxin, 1998] Ioan ROXIN, Nabil AKROUT Streaming Technology for Tele_education Via Satellite Proceedings of the IASTED International Conference Computers and Advanced Technology in Education May 27-30, 1998, Cancun Mexico

[Shammari, 1998 a] A.Al-Shammari, A.Karmouch On-Demand Multimedia Courseware Delivery over the Network Proceedings INDC'98 - 7th IFIP/ICCC Conference on Information Networks and Data Communications, Portugal, June15-17, 1998

[Shammari, 1998 b] A.Al-Shammari, A.Karmouch Multimedia Interactive Telelearning Prototype Proceedings of International Conference. on Computers and Advanced Technology in Education (CATE'98), Cancun, May 27-30, 1998

[Eggebraaten, 1997] Tom Eggebraaten Distance Education via the Internet Formal Research Paper 11/24/96 http://www.cs.und.edu/~eggebraa/distance.html

[JavaSvr, 1998] Java Web Server
http://jserv.java.sun.com/products/webserver/index.html

[Shirmohammadi, 1997] Shervin Shirmohammadi, Nicolas D. Georganas JETS: a Java-Enabled TeleCollaboration System Proceedings of IEEE International Conference on Multimedia Computing and System p. 541-547, June 1997

[Sun, 1997] Sun Microsystems The Java Language: An Overview
http://java.sun.com/docs/java/java-overview-1.html

[ObjectStore, 1997] ObjectStore Java API ObjectStore document
Http://www.objectstore.com

[Poon, 1997] Nelson Poon, A. Karmouch 3 Dimensional Multimedia Interactive Courseware Authoring in TeleLearning System Proc. of IEEE Canadian Conf. Elec. & Computer Engineering, St.John's, NFLD.

[Wang,1997] Ruiping Wang, A. Karmouch Multimedia Courseware Delivery over Broadband Networks, IEEE International Conference on Communication 97, June 8, 1997

[Wang, 1996] RuiPing Wang , A. Karmouch A Broadband Multimedia TeleLearning System Proceedings of IEEE Canadian Conference. on Electrical and Computer Engineering '96

[NetMeeting, 1998] Microsoft NetMeeting document http://microsoft.com/netmeeting/

[JSDK, 1997] Java Servlet Development Kit. http://java.sun.com:80/products/java-server/servlets/index.html#sdk

[Cassady-dorion, 1997] Luke Cassady-Dorion, et al, Industrial Strength Java, ISBN 1-56205-634-4 New Riders Publishing 201 west 103rd Street Indianapolis, IN 46290 USA

[Aunff, 1996] Ed Anuff, JAVA Sourse book : A Complete Guide to Creating Java Applets for the Web, ISBN 0-471-14859-8

[Booch, 1991] Grady Booch, Oject-Oriented Design, Redwood city, Calf. : Benjamin/Cummings.

[Venhelsuwe, 1997] Laurence Venhelsuwe, Ivan Phillips, et al, Mastering Java1.1, second edition, ISBN 0-7021-2070-9 SYBEX inc. , 1151 Marina Village Parkway, Alameda CA 94501