Integrating the Moodle Course Management System into a Collaborative Virtual Environment

Project Report
Submitted in
Partial Fulfillment for the Degree of
Master's in Computer Science

by Rashmi S Ramagiri
under the guidance of

Dr. Clinton Jeffery
Assistant Professor, Department of Computer Science

Department of Computer Science
New Mexico State University
Las Cruces, NM 88003

Table of Contents

Abstract

A virtual environment that acts as a user interface to provide communication between human-human in addition to human-machine is called Collaborative Virtual Environment. An Educational Virtual Environment is a collection of integrated tools enabling online learning by providing a delivery mechanism which involves tracking of student activity along with access to resources virtually. Educational CVEs are a technological breakthrough with the potential to provide students, teachers and learners a platform for discussion and sharing of views. CVEs have tremendous potential to enhance the learning environments of the future; they provide a real time environment for users to interact when they are physically in two different geographical locations. The instructors can operate courses remotely and the students can access the course materials through the Collaborative Virtual Environment. This project is an attempt to provide educational/courseware support for instructors and students in the Collaborative Virtual Environment. The design includes the integration of features of Moodle *(Modular Object-Oriented Dynamic Learning Environment),* a free, open source course management system for online learning. This involves integration of a MySql relational database which allows courseware support to the Virtual Environment called *unicron*. This project is written in the Unicon language which manipulates SQL databases through the Open DataBase Connectivity connection mechanism.

.

1. Introduction

Virtual Reality (VR) sprang onto the public stage in the 1980's due to media interest and related works of science fiction. VR promised to change the way people experience and interact with computers. One of the major fields in VR is Collaborative Virtual Environments (CVEs). The main goal of CVEs is to develop better and more effective ways to use computers for communication [Churchill].

With the advent of online learning through the Internet, a new breed of educational environment has emerged rapidly, wherein individuals can share information through remote interaction with each other. Students can collaborate to learn, solve problems and be highly productive and efficient. A Collaborative Virtual Environment is a computer-based, distributed, virtual space where people can meet and interact with others. Collaborative Virtual Environments can be used productively in the field of education, as a potential technology to facilitate more active student and instructor collaboration and learning. Collaborative Virtual Environments help to provide distance education and are effective substitutes for bonding people and for in-person social necessities. Collaborative Virtual Environments are of particular interest to researchers and students in areas related to computer supported cooperative and collaborative work and human computer interaction. They allow the instructor to teach courses at their own technological comfort level by providing templates for course management.

Though existing tool such as Centra, provides whiteboard, handraising, webpages sharing/co-browsing. It does not offer the advantages of educational CVE which include nearness and social presence through a shared virtual space populated by avatars. A student should get the real class room experience even while accessing the courses remotely [Jeffery05]. A research team at NMSU has developed Unicron, a Collaborative Virtual Environment for the purpose of computer science distance education that brings a class room experience to students in remote locations. Unicron has been modeled after the first floor of Science Hall at NMSU. Unicron will provide distance education which will supersede the traditional video-conferencing and use of educational tools such as WebCT by integrating these existing educational technologies into the 3D virtual community.

This report presents the design and implementation of the courseware support in a collaborative virtual environment allowing remote student-instructor interactions. This is achieved by integrating the features of open-source course management software called Moodle that supports a social constructionist framework of education and allows educators to create an effective, productive online learning environment, hence making online learning more.

The project was developed on Linux (2.6.11.4-21.9-SMP) running on a Pentium 4 processor running at 3 Giga hertz with HyperThreading. The memory of the system was 512MB. The version of the unicon compiler used to build the project was unicon version 11.3(beta). The project required a few other packages, they are mysql-4.1.10a-3 and
unixODBC-2.2.10-3.

2. Unicon ODBC

The term Unicon stems from Icon [Jeffery03]. Unicon is platform-independent, portable, robust language and is well-suited in developing object oriented applications, network centric applications, and database programming.

Database Architectures and ODBC
Unicon supports databases such as DBM and MySQL. A connectivity mechanism is needed in order to communicate with databases such as MySQL etc. The ODBC interface serves this purpose. Unicon's SQL tools require numerous software components. Firstly, a SQL server is required for ODBC connectivity, along with a generic account with suitable privileges on the SQL server is required for connectivity. Lastly the ODBC driver manager and an ODBC driver configured to suit the database requirements are needed on the clients. The next task is to code the Unicon client in order to connect to the database [Balbi02].

Opening a SQL Database
A sql database can be opened by an open () function is used to open a connection to a SQL database. For the integration of moodle into unicron, the database operations were performed on moodle database. For more information about unicon ODBC, please refer to [Balbi02].

An example of the open () function that connects to the database is shown below.
C:= open ("cve", "o","moodle", "username", "password")

Querying and Modifying a SQL Database
The sql() function is used to perform queries on the database[Balbi02]. The two arguments are the database and a sql command such as a select, update, modify etc. The sql function places the cursor at the beginning row of the selected result set.

One example of the sql function is.
sql (C, "select firstname from person where firstname like 'Andr %'")
The second argument in the above sql call returns rows that match the criteria of all names starting with Andr.

Navigating across the selected rows in a table
The fetch() function is called in order to navigate across the resultset (the returned rows). The fetch() function takes the database name as its argument. The return value of this function is a single row in a table.

If a fetch (C) returns a row containing column firstname, one can write
row: = fetch (C)
write (row.firstname)
When the fetch function has one argument, fetch (C) moves the cursor forward one position.
When passed with two arguments, the database name and the column, fetch (C, firstname) returns a single column from the current row.

SQL Types and Unicon Types
Some of the data types supported by SQL and Unicon are CHAR and VARCHAR that correspond to Icon strings, INTEGER and SMALLINT data types that correspond to integers, FLOAT and REAL that correspond to reals etc. The conversion between SQL and Icon is fairly easy and with minimal changes to the data format. One needs to understand the pros and cons related to SQL implementations.

3. Moodle

Moodle is an open source web-based course management system designed based on the principles of social pedagogy. Moodle produces Internet-based courses and websites; it requires a web server with integrated PHP (which is the scripting language) and database support [moodle].

Moodle supplements the traditional face-to-face learning by providing online-classes, is user-friendly, simple, efficient, compatible, easy to install on any platform that supports PHP. It requires an underlying Sql database.
The entire moodle system is managed by an admin user during setup. The administrator is permitted to customize the site colors, fonts, layout etc to suit his needs. The Moodle is then added with the required activity modules. The source code is easy to modify to suit the student's needs.

3.1 Moodle Modules

This section discusses the different modules supported by moodle.

3.1.1 Site Management

This module is used for site management used for web interface which has no importance for unicron. This application incorporates authentication mechanisms using plug-in modules, which allows the legacy systems to be easily integrated. The standard email mechanism enables students to create their own login accounts which are verifiable by confirmation. The LDAP mechanism facilitates login accounts to be checked against an LDAP server. The administrator can specify which fields to use. Typically the IMAP, POP3 and NNTP are checked against mail or news server and security provisions such as SSL certificates and TLS are supported as well. This application also supports external authentication wherein any database with at least two fields can serve as an external authentication source.

3.1.2 User Account Management

Though this module is important for user account management, its not been incorporated in unicron yet, all the user account management is done through the web interface provided by moodle. Different sets of accounts can be created on the server. The admin account has administrative privileges and controls course creation and user accounts. Every individual has one account assigned for the entire server; however access privileges may vary for these accounts. A course creator account can be assigned privileges to create courses and teachers typically have editing privileges which can be revoked to prevent course modification. Teachers can also enroll and unenroll students manually. This process can however be automated. Security is incorporated through key mechanism such as enrollment key to prevent non-students from accessing the course page. Time-zone data and language information can also be accessed by the user.

3.1.3 Course Management

A full teacher has admin privileges to restrict other teachers and also to control the over-all settings for a course. The teacher can set the course formats by week, by topic or by social format. An array of course activities such as Quizzes, Forums, resources, assignments, etc can be developed. The course catalog homepage could be used to display any changes to the courses. The webpage can also be used to post grades for quizzes and assignments. These web pages can be edited using a web-based programming editor such as an HTML editor. Activity reports can also be generated for logging and tracking access information. Graphs and visual information can be embedded in these reports.

3.1.4 Assignment module

This module has been completed integrated into the unicron. It is used to post assignments with due dates wherein students can upload their assignments. It also provides a provision to timestamp student submissions and to display grades.

3.1.5 Chat module

This module has already been integrated in unicron in the form of text-based communication. It is unlikely that, this module will be integrated in the future. But the logging information is used to the full extent by logging in all the chat information for archiving purpose.

3.1.6 Choice Module

The Choice Module uses the principle of polls to obtain student feedback through votes. It also supports graphical data. This module has not been integrated into unicron.

3.1.7 Forum Module

This module has been partially integrated. It enables discussions relevant to teachers, courses and students. Discussions can be single or multi-threaded. The module also supports images and also allows discussion threads to be moved from one forum to another.

3.1.8 Quiz Module

The quiz module relies on a database of questions to generate a quiz for every student. Questions can be sequential or random. The module allows quizzes to be created automatically and inserting the time-frame for each quiz. At the teacher's discretion, quizzes can be set for students to attend multiple times and can also include images, true or false questions, short- answer questions, embedded answer questions, etc.

3.1.9 Resource Module

This module has been partially integrated, only text information are displayed. The module supports interactive content such as video, sounds, PowerPoint, flash, etc. It can also be seamlessly linked with external applications.

3.1.10 Survey Module

The survey module supports built-in surveys with graphs and spreadsheets that can be posted on a web-page. The module also provides feedbacks to students once completed. This module is not integrated into the unicron.

3.1.11 Workshop Module

This module serves as an assessment tool wherein peers can assess documents and teachers can grade their assessed documents. It supports a wide range of grading scales such as the likart scale. This module is not integrated into the unicron.

3.2 Working with the Moodle Interface

This section deals with adding a course from the instructor's point of view and course navigation from the student's side using the moodle web interface.

3.2.1 Adding a course in Moodle- Teacher's Guide

Teachers can create online courses with moodle. The teacher logs in via the web with the teacher account assigned by the administrator. Under the administration icon on the course home page, the option setting allows the teacher to change the course settings ranging from its name to what day it starts.

Under the settings option the course format will help the teacher to use the basic layout of the course like a template. Moodle supports three formats: weekly format, topics format and social format. Weekly format covers lectures exactly for one week whereas the topic format covers any topic the teacher likes. The social format is based around just one forum which is displayed on the main page and does not use much content.

Moodle also supports uploading all kinds of files such as web pages, audio files, video files, word documents, etc. This can be done through the Files link under the administration icon. All the uploaded files are stored on the server and these files can be moved, renamed, deleted or edited. These files are accessible to the teachers alone and are made available to the students later. The files are organized into sub-directories for easy and convenient accessibility. In the current moodle, only one file can be uploaded at a time through the web interface. In order to upload more than one file, a zip program can be used to compress and group the files into an archive file which can then be uploaded.

In the main course page, the teacher can add all the course activity modules in the order that students will access them. The standard course activities included are assignment, choice, forum, resource, quiz and survey. In order to add a new activity the teacher has to go into one of the format sections explained above and select the type of activity from the pop-up menu.

3.2.2 Course Navigation- Student's Guide

The student has to first access the course website using the site address provided by the teacher. If the student is accessing the course website for the first time, he has to start by first creating a new account. The new account created will give the student access to all the courses. Individual courses may require a once only enrollment key to access them.

The procedure to create a new account starts with the student clicking on "Start now by creating a new account" button in www.moodle.org page followed by filling out the new account form and creating a username and password. A confirmation email containing a web link will be sent to the email address provided by the student. The student account will be confirmed once the student clicks on the web link provided in the email. He can now see all the available courses and can select the course that he wants to enroll. When a student tries to enroll into a course for the first time, he

will be prompted for the enrollment key which will later be used by this student to enter the respective courses.
Once the student enters the course he can navigate through the various pages using the course activity modules.

3.3 Installing Moodle on Linux

Moodle is primarily developed in Linux using Apache, MySQL and PHP [moodle]. It requires

1. Web server software. It works under any web server that supports PHP, such as IIS on Windows platforms.
2. PHP scripting language (version 4.1.0 or later).
3. a working database server such as MySQL

Moodle can be installed by downloading and copying the files into the main web server documents directory. The install script is then run to create config.php which can be done by accessing the Moodle main URL using a web browser or by accessing http://yourserver/install.php directly. The web server is set up to use index.php as a default page. This is done by using a DirectoryIndex parameter in the httpd.conf file in Apache. Moodle requires a number of PHP settings to be active in order for it to work. On most servers these will already be the default settings.

Creating a database called moodle

An empty database is created in the database system along with a special user that has access to the database and is granted administrative privileges.

Some examples for Unix command lines for MySQL are:

```
# mysql -u root -p
> CREATE DATABASE moodle;
 > GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, INDEX, ALTER ON moodle.*
       TO moodleuser@localhost IDENTIFIED BY 'yourpassword';
> quit
```

Creating a data directory called moodledata

A data directory is created by the moodle installer that needs some space on the server's hard disk to store uploaded files such as documents and user pictures. This can be done manually if it fails.

Configuration

When the admin page is accessed for the first time, a GPL agreement is provided that must be agreed upon by the user to continue with the setup. Moodle will then set up the database and create tables to store data, followed by a number of SQL statements and status messages. It then allows the users to define the parameters for the Moodle site. The final step is to create top-level administration user to the admin pages. The admin then can perform tasks such as creation and deletion of courses, creation and editing of user accounts (student and teacher accounts), etc.

Set up cron
Moodle modules sometimes require continual checks to perform tasks. The admin directory contains a script called cron.php that does the continual checks. These checks cannot run by themselves and need a setup mechanism wherein the script is run at regular intervals. This setup mechanism is known as the cron service.

Using the crontab program on Unix
All that Cpanel does is provide a web interface to a Unix utility known as crontab.
Type the following in the command line
crontab -e
and then add one of the below commands using any editor tool:
*/5 * * * * wget -q -O /dev/null http://example.com/moodle/admin/cron.php

3.4 Configuring unixODBC

The following section helps in configuring unixODBC on the client side.

Fetching ODBC driver
The unixODBC source distribution is available at www.unixodbc.org, a gzipped tar file which is uncompressed using the gunzip command and then the resultant file is untarred using the unix tar (tape archive retrieval) [Easysoft].
For example
gunzip unixODBC-2.2.12.tar.gz
tar -xvf unixODBC-2.2.12.tar
Change into the resultant directory and run:
. /configure --help

Installing ODBC driver
Installing ODBC driver can be done three different ways. First Method is by writing a program which links with libodbcinst.so and calls SQLInstallDriver. Second Method is by creating an ODBC driver template file and running odbcinst. For example, `odbcinst -f template_file -d -I`, where the template file must contain the Driver and Description attributes. Third method is by editing the odbcinst.ini file and adding the driver definition. Each driver definition in the odbcinst.ini begins with the driver name in square brackets which is followed by Driver and Setup attributes. Driver is the path to the ODBC driver shared object and Setup is the path to the ODBC driver setup library
In unixODBC ODBC drivers are defined in the odbcinst.ini file.

File DSN's
ODBC has a file DSN that stores the connection information in a file saved on to a central server that is accessible to all the workstations.
Configuring a MyODBC DSN on *nix.

odbcinst.ini
This contains a section heading that provides a name for the driver. The Driver and Setup paths point to the ODBC driver and setup libs.

The template for odbcinst.ini is:

```
[MySQL]
Description    = ODBC Driver for MySQL
Driver         = /usr/lib/unixODBC/libmyodbc3.so
Setup          = /usr/lib/unixODBC/libodbcmyS.so
FileUsage = 1
CPTimeout =
CPReuse   =
UsageCount     = 2
```

[.]odbc.ini

The contents of the odbc.ini files follow just the same format as the odbcinst.ini entries.
The template for odbc.ini is:

```
[mydsn]
Description = MySQL database moodle
Driver = MySQL
SERVER = cve.cs.nmsu.edu
DATABASE = moodle
PORT = 3306
Socket =
Option =
Stmt =
USER = <username in mysql>
password = <password for the above user in mysql>
Trace = 1
TraceFile = error.log
```

The template for odbc.ini in /etc/unixODBC/odbc.ini is:

```
 [mydsn]
Description = MySQL database moodle
Driver = MySQL
Server = cve.cs.nmsu.edu
Database = moodle
Port = 3306
Socket =
Option =
Stmt =
USER = <username in mysql>
password = <password for the above user in mysql>
```

Parameter	Default Value	Comment
Server	Localhost	The hostname of the MySQL server.
Database		The default database.

Option	0	Options that specify how MyODBC should work. See below.
Port	3306	The TCP/IP port to use if server is not localhost.
Stmt		A statement to execute when connecting to MySQL.
Password		The password for the user account on server.
Socket		The Unix socket file or Windows named pipe to connect to if server is localhost

Establishing a Remote Connection to Server from System A

Server side:

- Start the MySQL server.
- Use grant to set up an account with a username and a password of the user who can connect from system B

  GRANT ALL ON *.* to 'myuser'@'A' IDENTIFIED BY 'mypassword';
- The GRANT statement grants all privileges to user myuser for connecting from system A using the password mypassword. To execute this statement, root privileges are required on system A.

System A side:

Configure a MyODBC DSN using the following connection parameters:

- DSN = remote_test
- SERVER or HOST = A (or IP address of system A)
- DATABASE = test
- USER = myuser
- PASSWORD = mypassword.

4. Requirements

This section discusses about the functional requirements and the non-functional requirements. Certain features are taken from moodle design and integrated into unicron which allows integration of courseware support to the CVE. These features include addition of assignment submission, display of lecture notes and assignment description posted for a particular course. Also display the forums or posts posted to a particular course in the unicron. Chat module from moodle is not integrated into unicron, since it is already a part of unicron mode of communication. To keep track of student's activity, all the client information is logged including chat information.

4.1. Functional Requirements

View Students information

User: Student

Description: View list of courses enrolled by the user. The following details will be visible

- Individual Name
- The course list
- The instructor's name for the course
- The instructor's email address for the course

Dependencies/Constraints: None

View Courses Enrolled
User: Student
Description: A list of all possible courses the student is enrolled. The following details will also be visible

- The course Name
- The Lectures posted for the course
- The assignments posted for that course
- All the postings for the course

Dependencies/Constraints: None

View Assignments for a course
User: Student
Description: A list of all the assignments posted for a course. The following details will also be visible

- The name of the assignment
- Complete information of the assignment
- Due date for the assignment
- The time of posting of the assignment
- The time of last submission by the user
- A dialogue box for uploading the assignment

Dependencies/Constraints: If the due date for the assignment is over, no submission is possible for that particular assignment. The dialogue box for uploading the assignment will not be shown to the user.

Select Lecture Topic
User: Student
Description: Students may select topic from a dropdown list

- Complete information of the Lecture notes selected by the user

Dependencies/Constraints: None

4.2 Non-Functional Requirements

This section includes the hardware and software constraints.

4.2.1 Hardware Constraints and specification

The hardware constraints and specifications on the server side include a minimum hardware support to host Unicron and Moodle. On the user side, any computer system that can run Unicron and a network connection, in case the unicron server or moodle database are hosted remotely

4.2.2 Software Requirements

To run unicron on the server side, the system should include unicron server and a mysql database server. The unicron server and mysql database server need not be on the same machine. This project was tested on Linux platform, and is recommended to run on Linux platform, till further testing is done on other platforms like Windows, or any other operating system that Unicron supports.
On the Client side, Linux operating system is recommended with unicron client installed on it. Client side should contain MyODBC-unixODBC package for Unicron-moodle communication.

5. Design

The main goal of this project is to integrate the features of moodle into unicron. This is achieved by providing a thin layer between Sql and the cve. The connection between the cve and the database is created in the class moodle. To provide course management the above connection helps in executing the sql operations between the cve and the database. The course management is mainly focused in displaying the lecture information, assignment description, assignment submission, logging chat information and user activity. The implementation details of the above mentioned class is discussed in the later section.

When a user logs in, apart from creating a virtual environment, a connection is made to the MySQL database from the client side to retrieve the courses the user is enrolled to. The CVE displays the course information like the assignments, lecture notes and posting. The user can also submit the assignments from the Virtual Environment.

In order to achieve the goal of this project, the design should involve developing a unified, intuitive and single application that would make use of collaborative virtual environment features and moodle. The developed interface should be clearly organized and easy to navigate. To provide this kind of interface, a new sub window is added along with the virtual environment, which would represent the courses that the user enrolled into.

To display the course information a new tab is created at runtime by the class CourseTabItem. This class is called from [illegible]lass which deals with the u[illegible]

The below figure is [illegible]m showing the inheritance [illegible] this project.

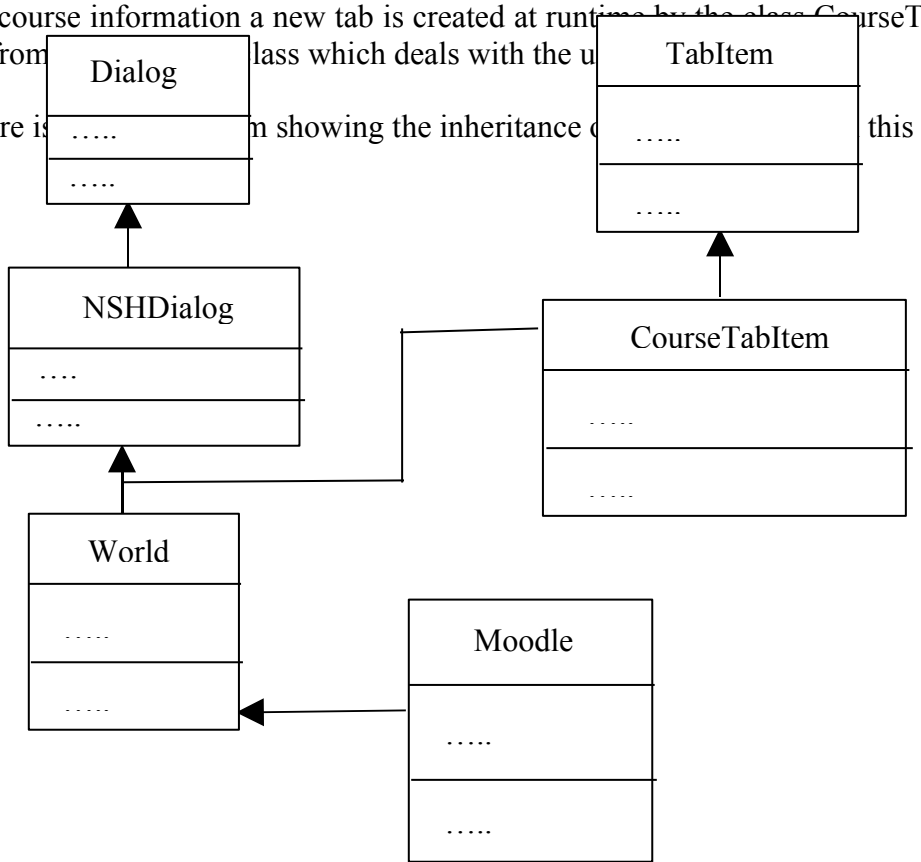


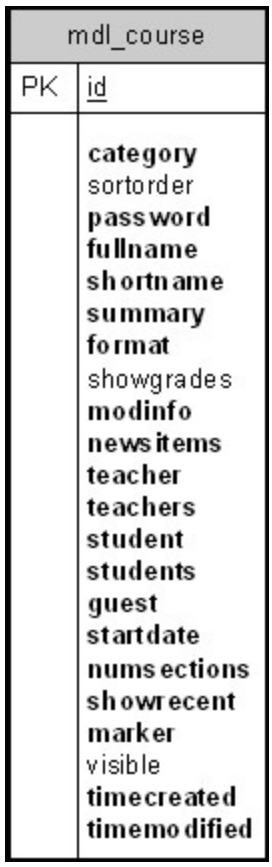
Figure 1: UML Diagram

Moodle Database Structure

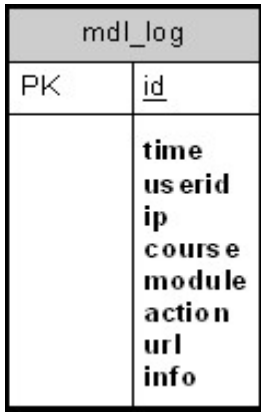
- Every table has an auto-incrementing id field (INT10) as primary index.
- The main table containing instances of each module has the same name as the module.
- Following are the minimum fields contained in the main table:
  id - as described above
  course - the id of the course that each instance belongs to
  name - the full name of each instance of the module
- Other associated tables with a module that contain information are named as school_info (example, mdl_user_students)
- Simple and short column names are used.
- Columns that contain a reference to the id field of another table (eg widget) are called as widgetid.
- Boolean fields are implemented as small integer fields (eg INT4).
- Most tables have a time modified field (INT10) which is updated with a current timestamp obtained with the PHP time () function.

There are 132 tables in the moodle database. The main twelve tables are mdl_config, mdl_course,mdl_course_categories,mdl_course_modules, mdl_course_sections, mdl_log, mdl_log_display,mdl_modules,mdl_user,mdl_user_admins,mdl_user_students,mdl_user_teachers. These tables are called as the meta-tables. The other tables are listed in the appendix section of this

document. The tables that are used in this project are listed below: mdl_user, mdl_course, mdl_resource, mdl_forum, mdl_user_students, mdl_assignment, mdl_assignment_submissions, mdl_log, mdl_chat_messages, mdl_chat_users and mdl_user_teachers.



The mdl_course table has 32 fields of which only a few are used, the rest of them are for future use. This table is a read-only table which is used to retrieve the course enrolled by the student and for the information pertaining to a course which is displayed in the cve. The fields used read from the cve are id, fullname, shortname and summary. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10.Fullname is the name of the course which is of type varchar, the maximum size of it is 254 characters and cannot be null. The shortname is the call number of the course of type varchar, the maximum size of it is 15 characters and cannot be null. The summary is the description of the course which is of type text and this field can hold data up to 4 GB and cannot be null.



The fields of this table are written from the cve, it logs the user activity from the cve. The fields that are written to mdl_log table are id, time, userid, ip, course and action. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. The time represents the time at which a particular action was performed by the user. It is of type unsigned int and ranges from 0 to 10. It stores the epoch time and the default value is 0. The userid is the id of the user who performs the action and is of type unsigned int. The values range from 0 to 10 and default value is 0. The IP represents the IP address of the client machine where a particular action was performed and is of type varchar with a maximum size of 15 characters. It cannot be null.

mdl_chat_messages	
PK	id
	chatid userid system message timestamp

The fields of this table are written from the cve, this records the chat messages send from the user in the cve. The fields that are written to mdl_chat_messages table are id, userid, chatid, message and timestamp. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. The timestamp represents the time at which a particular message was sent by the user which is of type unsigned int. It stores the epoch time with values ranging from 0 to 10 and default value being 0. The userid is the id of the user who performs the action and is of type unsigned int. The values range from 0 to 10 and default value is 0. The chatid represents the chat id of the user which is different from the userid. It is of type int with values ranging from 0 to 10 and the default value being 0. Message is of type text which stores the conversation of a particular user.

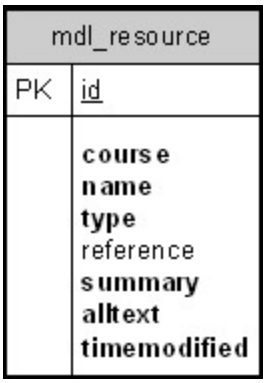
.

mdl_chat_users	
PK	id
	chatid userid version ip firstping lastping lastmessageping sid

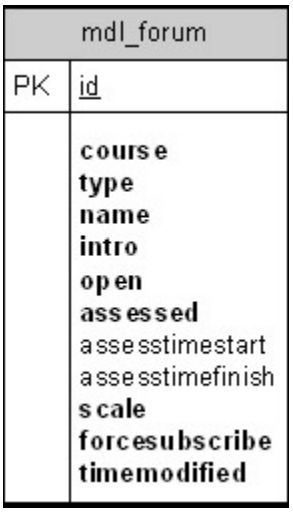
This table is a read-only table which is used to keep track of the users who want to communicate with the others. The fields used from mdl_chat_users table are id, userid, chatid and ip. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. The userid is the id of the user who performs the action and is of type unsigned int. The values range from 0 to 10 and default value is 0. . The chatid represents the chat id of the user which is different from the userid. It is of type int with values ranging from 0 to 10 and the default value being 0. The IP represents the IP address of the client machine where a particular action was performed and is of type varchar with a maximum size of 15 characters. It cannot be null.

mdl_assignment	
PK	id
	course name description format resubmit type maxbytes timedue grade timemodified

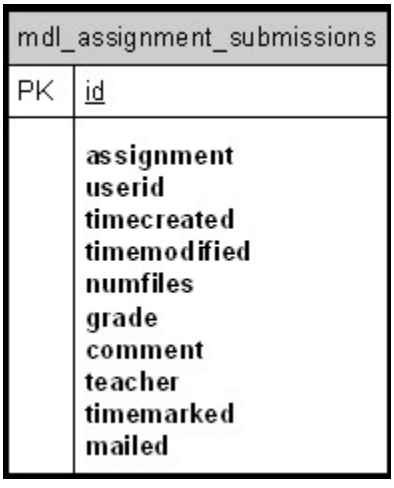
This table is a read-only table which is used to retrieve the assignment information when an assignment is selected. It displays the description of the assignment selected. The fields used from mdl_assignment table are id, course, name, description, timedue, timeavailable, preventlate, timemodified and resubmit. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. Course represents the course id number of type unsigned int with values ranging from 0 to 10 and the default value being 0. Name is the name of the course of type varchar having a maximum size of 255 characters and cannot be null. Timedue represents the time when the assignment is due and is of type unsigned int with values ranging from 0 to 10, the default value being 0. The timeavailable represents the time when the assignment is available and is of type unsigned int. Timemodified represents the time when the assignment was last modified and is of type unsigned int with values ranging from 0 to 10, the default value being 0. The preventlate field does not allow late submissions. It is of type unsigned tinyint with values ranging from 0 to 2, the default value being 0. The resubmit field is of type unsigned tinyint of maximum size 2 and a default value of 0. This decides whether resubmission is possible for a particular assignment.



This table is a read-only table which is used to retrieve the resource files for a particular course and displays the description of the resource in the course tab of the cve. The fields used from mdl_resource table are id, course, name and alltext. . The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. Course represents the course id number of type unsigned int with values ranging from 0 to 10 and the default value being 0. Name is the name of the course of type varchar having a maximum size of 255 characters and cannot be null. Alltext represents the contents of the lecture notes and is of type text.



This table is a read-only table which is used to retrieve the postings for a particular course which is displayed in the course tab. The fields used from mdl_form table are id, course and name. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. Course represents the course id number of type unsigned int with values ranging from 0 to 10 and the default value being 0. Name is the name of the course of type varchar having a maximum size of 255 characters and cannot be null.



The fields of this table are written from the cve, this allows the student to submit their assignments from the cve. The fields that are written to mdl_assignment_submissions table are id, assignment, userid, timemodified, data1 and filename. The id is an auto increment field and is used as the primary key. It is of type unsigned int and ranges between 0 and 10. Assignment represents the assignment id number which is of type unsigned int. The values range from 0 to 10 with default value being 0. Timemodified represents the time when the assignment was last modified and is of type unsigned int with values ranging from 0 to 10, the default value being 0. data1 is of type longblob which stores the content of the submission.

6. Implementation

This section discusses the implementation of integration of moodle features into the unicron. In order to integrate courseware support into Unicron, user interface was changed accordingly to reflect the integration of moodle. Two new classes, Moodle and CourseTabItem were included. The gui screen shots are shown under Graphical Output section. The gui additions include addition of a course tab item and a course-user information section which gets created at run time. The created course tab item displays the lectures and assignments, if any pertains to the course selected by the student. Some methods were added into few existing classes and two new classes were added in order to get moodle integrated into Unicron, and are mentioned in the section below.

6.1. Moodle.icn

The moodle.icn file is present in unicron/src/client. This is a new class added into unicron. An object of this class is instantiated in nshworld.icn, which establishes a database connection between unicron and moodle. The Moodle class contains the following attributes and methods.

Attributes

1. ***uid:*** The uid defines the username of the person who wishes to connect to the Moodle database.

2. ***pwd :*** The pwd defines the password of the person who wishes to connect to the Moodle database. The password is stored as a binary string of 32 bit hexadecimal digits, using MD5 (Message digest algorithm)
3. ***moodle_db:*** Holds special handler to the Moodle database and can be used to perform file and table operations.

Methods

The Moodle class has three methods. They are

1. initialize_moodle (uID, Pwd): This method opens a database connection and returns a special handler. The input parameters for this method are username and password.
2. return_moodle_conn (): This method returns the special handler which is created during the open command of the database.
3. *initially (uID, Pwd):* This is the constructor method of the Moodle class. It sets up the database connection using the username and password provided during the instantiation of the moodle class.
    - uID username to access the moodle database.
    - Pwd password to authenticate to the moodle database.

In order to authenticate a user, she must provide a username and password. Proper permissions should be provided to the user to access the Moodle database.

The Moodle class helps in establishing connection between the moodle database and the unicron server/client. In order to establish a connection, a username and password must be provided. This is

done through the instantiation of the class moodle. The provided values are used to open a connection by using the following unicon statement.

Moodle_db:= open ("cve", "o","moodle", uID, Pwd)

The return value of the open () is a special handler which can be accessed through the method return_moodle_conn (). The special handler is required to perform database operations.
Moodle makes use of unicon odbc connectivity in order to connect to the database. An entry must be present in the .odbc.ini file under the user home directory. If an entry is not found, then an entry is added with appropriate field set.

6.2. nsh-world.icn

This is an existing class present at unicron/src/client. This creates a virtual environment for the nmsu science hall. In order to incorporate moodle functionality into the world class, a moodle object (Moodle.icn) is created, which contains a reference to the moodle database. The object creation is done in setMoodle (uid, pwd) method. The parameters passed to this method are obtained from the login dialog of the unicron client from nsh.icn.

6.3. CourseTabItem.icn

This class helps in creating gui elements required to display course support at runtime. It is present at unicron/src/client. All the gui elements like labels, textlist, editable textlist, separators and buttons required to display lectures, assignments and labels which provide information for a course selected are created in this class. Method SetContent sets up the gui elements at runtime. The gui of the course selection tab item can be seen in figure 3.

6.4. nshdlg.icn

This is an existing class present at unicron/src/client. This class hosts the user interface of the virtual environment and has the most modification to incorporate moodle into Unicron. Few methods were added into nshdlg.icn, which include

WritetoDB : This method executes the SQL queries. It checks if there is a connection established with the moodle database each time a sql query is executed. If it fails then it would display an error message.

```
sql (C, sqlstmt) | write_to_chat_win ("Error executing sql query")
```

In the above sql statement, the `world.moodle.return_moodle_conn ()` is the connection to the database which can be abbreviated as c.

DisplayUsername: This method displays the username of the user who logged in, is fetched from the table mdl_user from the moodle database and is displayed on the course panel. If a user does not have an entry in the moodle database, it displays a default string called "Username".

```
sql(C, "select id, firstname, lastname from mdl_user where
 username='" ||   world.moodle.uid ||"'")
```

where `id` is the userid
`firstname` is the firstname of the user who logs in
`lastname` is the lastname of the user who logs in
`world.moodle.uid` is the username provided during the login setup

ClientUserActivity: This method writes the login information of the user who is logged into the moodle database under mdl_log table. The log table information consists of the userid of the user who logs in, IP address of the client machine, time of login and the action string, which is by default “Logged In”.

```
sql(C, "insert into mdl_log (userid, ip, time, action) values
 (" || userID_mdl || ",'" || clientIP || "'," || &now || ",'Logged in') ")
```

Where `userid` is the id of the user obtained from userID_mdl
`ip` is the IP address of the client machine obtained from clientIP
`time` is the time when the user logs-in obtained from &now
`action` is the action performed by the user like login.

on_label_1: This method allows the user is able to open a web browser by clicking the link in the course tab ‘click here to check your grades’ which points to www.moodle.org where he can check his grades online from the moodle web-interface. In the method, the system command launches the execution of firefox, a web browser which points to the URL www.moodle.org

```
system ("firefox "|| "www.moodle.org" )
```

on_exit: This methos inserts an entry into the mdl_log table when the user logs out. The log information contains the last course number the user viewed, userid of the user who logs in, IP address of the client machine, time of logout and the action string which is by default, “Logged Out”.

```
sql(C,"select id from mdl_course where shortname='" || s || "'" )
Where, id is the course id of the course.
shortname is the short code of the course.

sql(C,"select id from mdl_user where  username= '"||world.moodle.uid ||"' and
password=MD5 ('"|| world.moodle.pwd ||"')")
Where, world.moodle.uid is the username provided during the login setup.
world.moodle.pwd is the password provided during the login setup.

sql(C,"insert into mdl_log (userid, course, ip, time, action) values (" || s1 ||
"," || s2 || ",'" ||   clientIP || "',"||&now||", 'Logged Out')")
```

Where, `id` is the user id.
`course` is the course id number.
`ip` is the IP address of the client machine obtained from clientIP.
`time` is the time when the user logs-in obtained from &now.
`action` is the action performed by the user which is a message.

The first statement retrieves the course id from the mdl_course table by comparing the course’s shortname. The second statement retrieves the id of the user from mdl_user table by comparing the username and password provided during the login process. The password is stored in MD5 checksum format in the mdl_user table. The third statement makes use of the information obtained from the above two sql statements and adds an entry into mdl_log table along with the IP address acquired by the clientIP, the time at which the user logs out which is obtained by &now, which

produces the current time and the action which is a message “logged out” when an user logs out of the CVE.

on_chat: This method records all the chat messages for later use in the mdl_chat_messages table. Every user has their own chatid and groupid which is different from the userid, which is fetched from the mdl_chat_users. The log information which gets stored into the mdl_chat_messages table contains chatid, userid, groupid, message and the timestamp when the message was sent. If a user does not have moodle access, the messages sent by the user are not recorded in the mdl_chat_messages table but the user can communicate between their peers.

```
sql(C, "select chatid, groupid from mdl_chat_users where userid=" || userID_mdl
insert into mdl_chat_messages (chatid,groupid, userid, message, timestamp )
values ("||chatID||","||groupID||","||userID_mdl||",'"||input_line||"'," || &now
||")")
```

where `chatid` is the id for the chat session obtained from chatID
`groupid` is the id of the group to which the user belongs to obtained by groupID
`userid` is the user id obtained from userID_mdl
`message` is the message sent by the user to communicate.
`timestamp` is the time when the user send the message obtained from &now
`userID` is the username provided during the login setup

Here, the chat messages sent by the users are logged in the database by inserting the chatid, groupid, userid, message and the timestamp into the mdl_chat_messages table. The values of chatid and groupid are fetched from the table mdl_chat_users by comparing the userid of the user who is communicating. The messages are the chat messages that are sent across the network through the chat window and the time of the message sent is obtained by the &now which returns the current time.

on_assignment: This method is called upon selection of the course number by the student, a new tab is created during runtime which contains an assignment list, a lecture list and a forum list. These different lists display the assignments, lecture notes and new postings pertaining to the course selected by the student. When the student tries to check on any of the assignments pertaining to a course, then the description of the assignment is displayed in textbox in the course tab. The description of an assignment is displayed to the user when an assignment is selected which is obtained from mdl_assignment table. The description is retrieved from the database by comparing the name of the assignment in the assignment list.

```
sql(C,"select description from mdl_assignment where name='"|| s || "'")
```

where `description` is the assignment/lab posted for a course; name is the course number

on_courses_btn: This method allows the user click on the courses button in the course panel in the CVE which lists the courses enrolled by the student. In order to achieve this task, the user id is obtained from mdl_user table, which forms the criteria to get the list of courses enrolled by the student from the table mdl_user_students.

```
sql(C,"select id from mdl_user where username='" || world.userId || "'")
sql(C,"select course from mdl_user_students where userid = '" || n || "'")
sql(C,"select shortname from mdl_course where id='"|| L[i] || "'")
```

where `id` is the userid of the user who logs in.
`world.userId` is the username provided during the login setup
`course` is the course number of the courses.
`username` is the name of the user obtained from world.userId
`shortname` is the shortname of the course

The first sql statement retrieves the id of the user from mdl_user table by comparing the username. The userid obtained above retrieves the courses from the mdl_user_students table. The third statement makes use of the course id obtained to fetch the shortname of the course which is displayed on the course list.

repeat1: This method logs in the user activity such as clicking on courselist, etc into the mdl_log table. The log information contains the last course number the user viewed, userid of the user who logs in, IP address of the client machine, time of action and the action string "Course List".
The method has two parameters, id and msg where id is the userid and msg is the message sent by the user across the network.

```
sql(C,"select id from mdl_user where username='"||world.moodle.uid||"' and
password=MD5('"|| world.moodle.pwd ||"')")
sql(C,"insert into mdl_log (userid, course, ip, action, time) values (" || s1 ||
"," || id || ",'" || clientIP || "','"|| msg ||"',"|| &now||");")
```

where `userid` is the id of the user obtained from mdl_user
`course` is the course number on which the student performed some task
`ip` is the IP address of the client machine obtained from clientIP
`time` is the time when the user logs-in obtained from &now
`action` is the action performed by the user which is a message.
`world.moodle.uid` is the username provided during the login setup
`world.moodle.pwd` is the password provided during the login setup.

The first statement retrieves the id of the user from mdl_user table by comparing the username and password provided during the login process. The password is stored in MD5 checksum format in the mdl_user table. The second statement makes use of the information obtained from the above sql statements and adds an entry into mdl_log table along with the IP address acquired by the clientIP, the time at which the user does some task obtained by &now, which produces the current time and the action which is a message "Course List" when an user tries to access the courseware support in the CVE.

upload_file: This method allows a selected file to be uploaded to the moodle database. In order to upload a file into moodle first a file dialog box pops up which allows the user to browse through the file system and select a file. Upon selection of the file, the assignment number, the userid of the user who is submitting the homework, the filename, the content of the file, the time when the last submission was made by the user and the number of files that a user submitted for a particular

assignment is stored into the moodle database. The decision of whether the user can resubmit the homework or have a deadline for any particular homework is made by the instructor through the moodle internet interface where he first uploads the assignment. Hence the user is allowed to resubmit homework files for the same assignment or he is not allowed to submit an assignment after the deadline.

```
sql(C,"select id from mdl_user where username='"||world.moodle.uid||"' and
password=MD5 ('"|| world.moodle.pwd ||"')")
sql(C,"select id from mdl_assignment where name='"||s1||"'")


sql(C, "insert into mdl_assignment_submissions (assignment, userid, filename,
data1, timemodified, numfiles) values ('"|| s3 ||"','"|| s2||", '"||filename||"',
"||s||"," || fs.mtime ||"," || numfiles ||");")
```

where `username` is the name provided by the user during the login setup obtained from `world.moodle.uid`
`password` is the authentication code provided by the user during the login setup obtained from `world.moodle.pwd`
`id` is the assignment id number.
`userid` is the id of the user who is submitting the assignment.
`Filename` is the name of the file along with the fullpathname.
`Data1` is the content of the file
`Timemodified` is the last time a file is submitted by the user
`Numfiles` represents the number of files submitted for an assignment

on_lect_notelist: This method allows, upon the selection of the course into which a user has enrolled, a new tab is created on the main section of the client window which contains the list of lectures, assignment and forum posts for that particular course. Upon selection of the lectures from the lecture list, a detailed description about the lecture is displayed to the user. The lecture notes are saved in the database under mdl_resource. New lectures for a course can be added through the moodle web interface.
A lecture list can be either saved into the database or uploaded as a file. When a lecture note is uploaded as a file, its relative path is saved into the database with reference to file upload directory which is /srv/www/htdocs/moodledata/. If a lecture is saved into the database, it is written into mdl_resource.

```
sql(C,"select   mdl_resource.alltext   from   mdl_resource,   mdl_course   where
mdl_resource.name='"||s ||"' and mdl_course.shortname = '" || CurrentTabItem ()
.label || "'")
sql(C, "select course from mdl_resource where name='" ||s || "'" )
sql(C,"select reference from mdl_resource where name='" ||s || "'")
sql(C,"select summary from mdl_resource where name='" ||s || "'")
```

Where, the `summary` is the content of the lecture selected
mdl_course.shortname is the shortname for a course
mdl_resource.alltext is the name of the lectures

on_courselist: This method allows the user to view the course page in the CVE he selects the course number from the dropdown list which shows the courses he is enrolled in with the lecture notes, assignments, forum list. It also displays the instructor information like name and email address. The user activity of navigating the course tab is logged into the mdl_log table. The log information contains the last course the user navigated in the CVE, the userid of the user who logs in, IP address of the client machine, time of navigation and the action string which is by default, "CourseList".

```
sql(C, "select * from mdl_user_teachers where course=" || s)
sql(C, "select * from mdl_user where id=" || teacherid)
sql(C, "select distinct name from mdl_assignment where course='" || s || "'")
sql(C,"select distinct name from mdl_resource where course='" || s || "'")
sql(C,"select name from mdl_forum where course='" || s || "'")
sql(C, "select id from mdl_course where shortname='" || s || "'")
sql(C, "select id from mdl_user where username='"||world.moodle.uid||"' and
password=MD5 ('"|| world.moodle.pwd ||"')")
sql(C, "insert into mdl_log (userid, course, ip, action, time) values (" || s1
|| "," || s || ",'" || clientIP || "','"|| msg ||"',"|| &now||");")
```

where `id` is the teacher id of the course in the first sql statement
`shortname` is the short description/code of the course
`id` is the user obtained from world.moodle.uid
`ip` is the IP address of the client machine obtained from clientIP
`time` is the time when the user logs-in obtained from &now
`action` is the action performed by the user which is a message.
`world.moodle.uid` is the username provided during the login setup
`world.moodle.pwd` is the password provided during the login setup.

Here, the user has an option of selecting the course he wants to navigate by selecting the course from the dropdown list. Once the course is selected, the teacher's name and email address pertaining to that course is retrieved from the mdl_user_teachers table. It also retrieves the assignment distinct name, lecture names and name of the posting for that course from mdl_assignment, mdl_resource and mdl_forum tables. The id of the user from mdl_user table is retrieved by comparing the username and password provided during the login process. The password is stored in MD5 checksum format in the mdl_user table. The activity of course navigation by the user is entered into the mdl_log table along with the IP address acquired by the clientIP, the time at which the user navigates obtained by &now and the action which is a message.

fetchIP: This method grabs the IP address for the client machine. The clientIP is required in order to record the log information which is obtained from the server side. Though the IP can be obtained from the client side, it's not done so due various factors like firewall, NAT or the client might masquerade an IP. In order to fetch the client IP, a message is sent from the client to the server requesting for the client IP. Upon the receipt of the request on the server side, the server sends back the client it's IP.

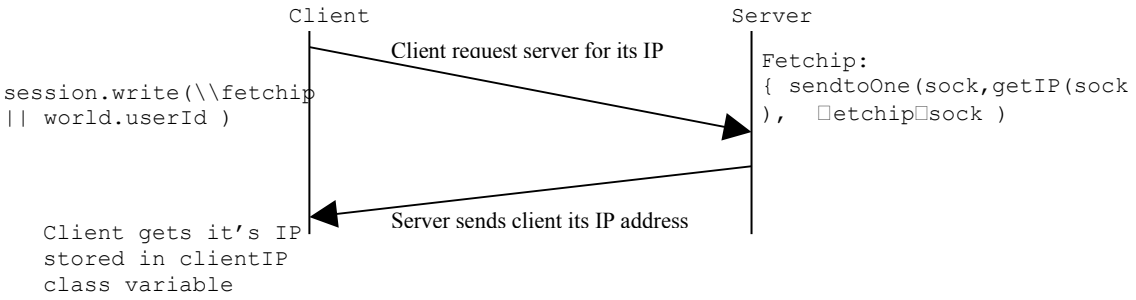


Figure 1: Client requesting the server for IP address

SetContentTab: This method calls the method setcontents in CourseTabItem,and passes parameters that contain the text to be displayed to the user. It also sets the events for the gui elements present in coursetabitem like SELECTION_CHANGED_EVENT, MOUSE_PRESS_EVENT, ACTION_EVENT.

7. Graphical Outputs

Below are a few screen shots from Unicron with courseware support.

NMSU Science Hall 3D-View

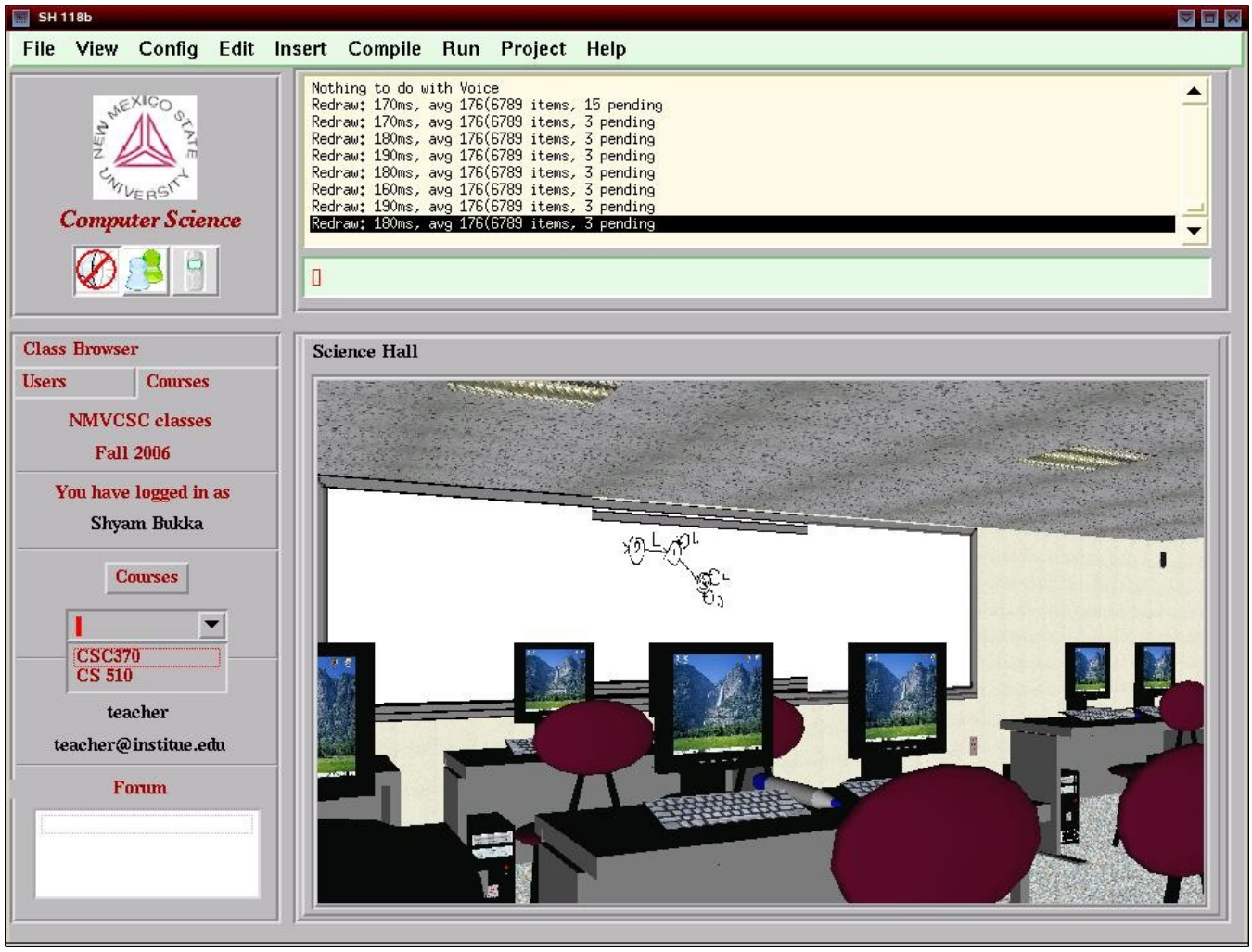


Figure 1: NMSU Science Hall 3D-View with moodle courseware support

The user is assigned a username and password to login into the NMSU virtual community. The user may not have access to a set of features like course support. The above screen shot represents the NMSU virtual CS department as seen from SH 118b classroom. It displays the courses in which the user is enrolled, showing some of the integration of unicron with moodle.

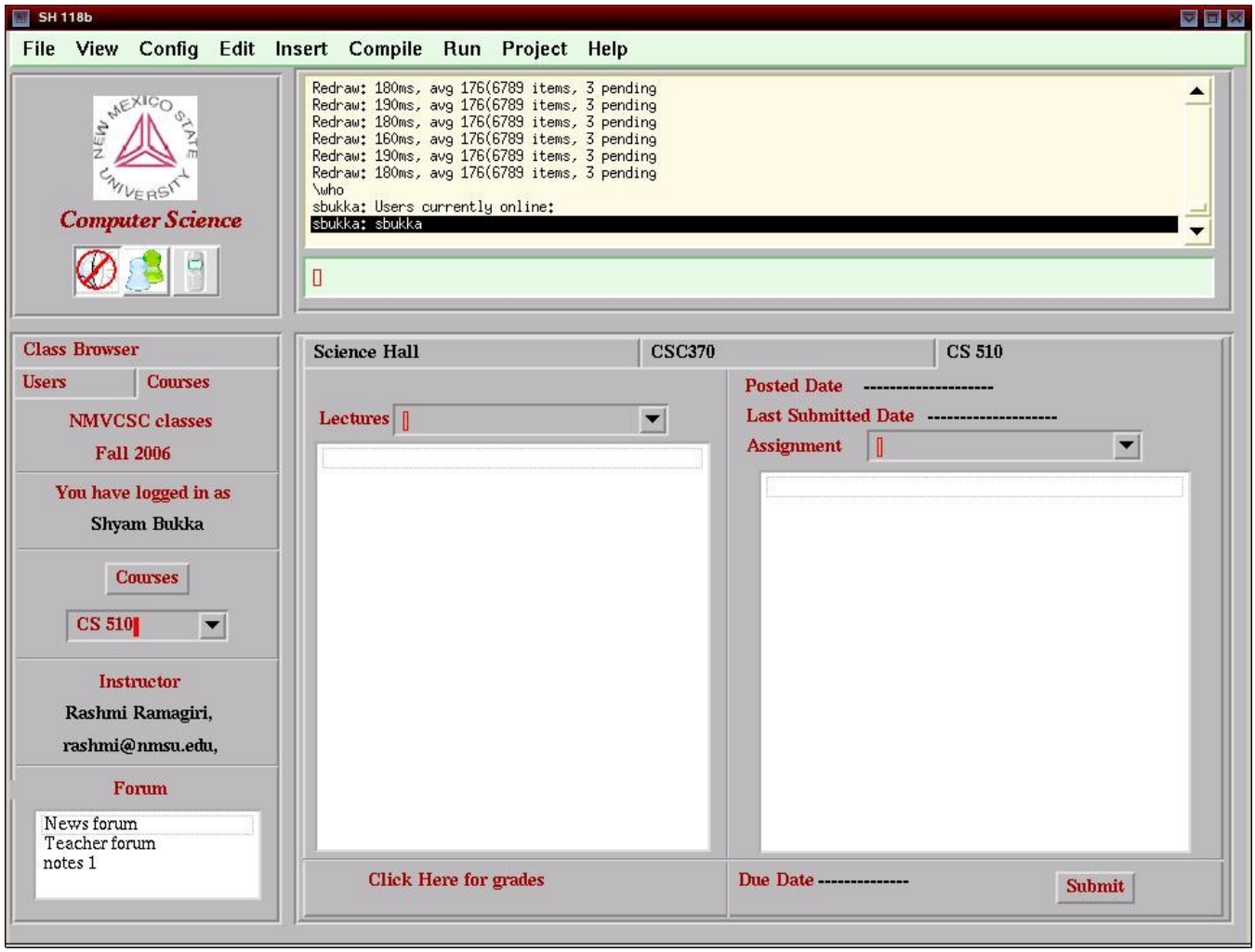


Figure 2: The course page with course details

The above figure shows the lectures, assignments and forum posts for the selected course. It also displays the instructor details like name and email address, if the user has to contact them.

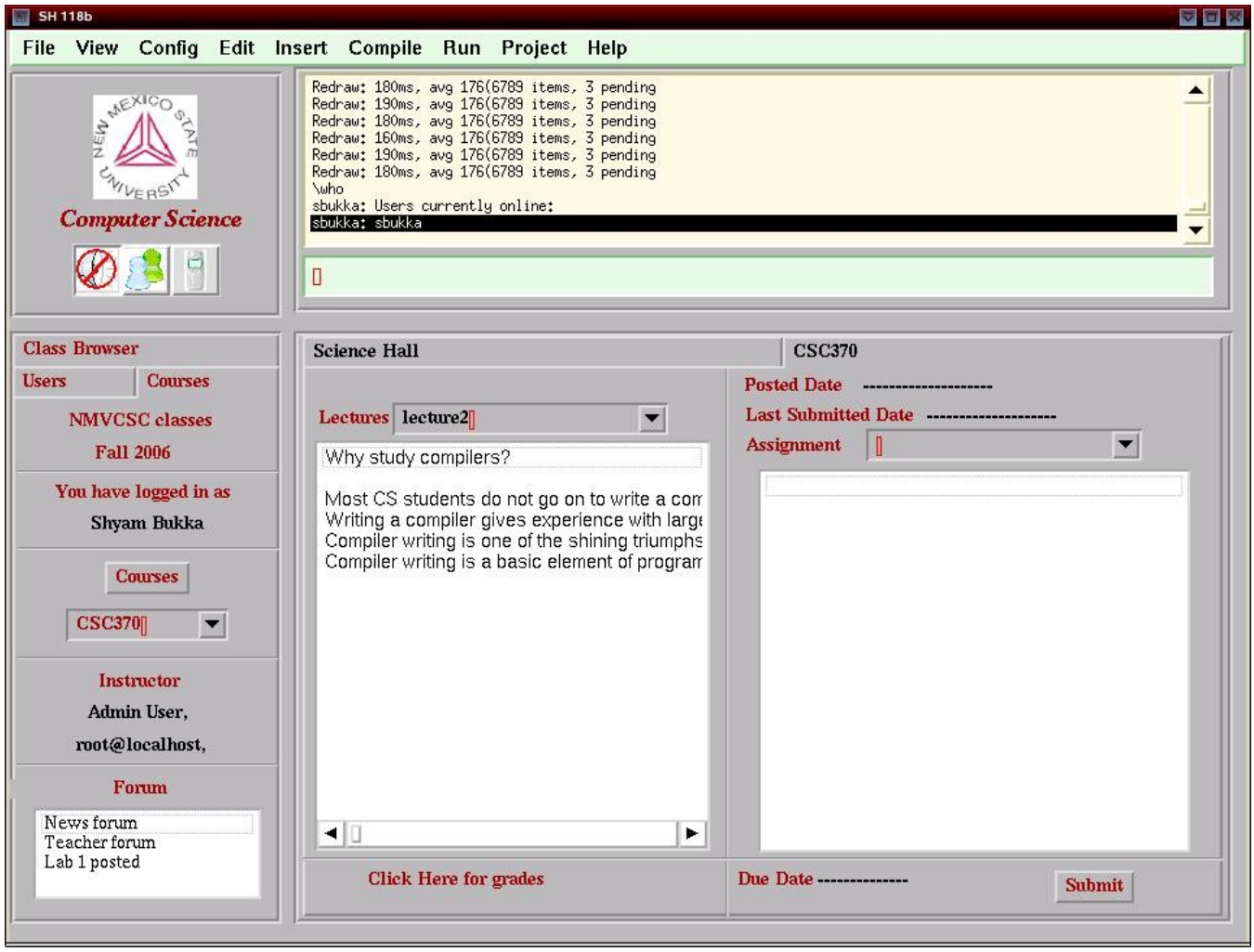


Figure 3: The Lecture Notes displayed in the course tab.

In the above picture, upon selection for a lecture for a particular course, lecture notes are displayed to the user

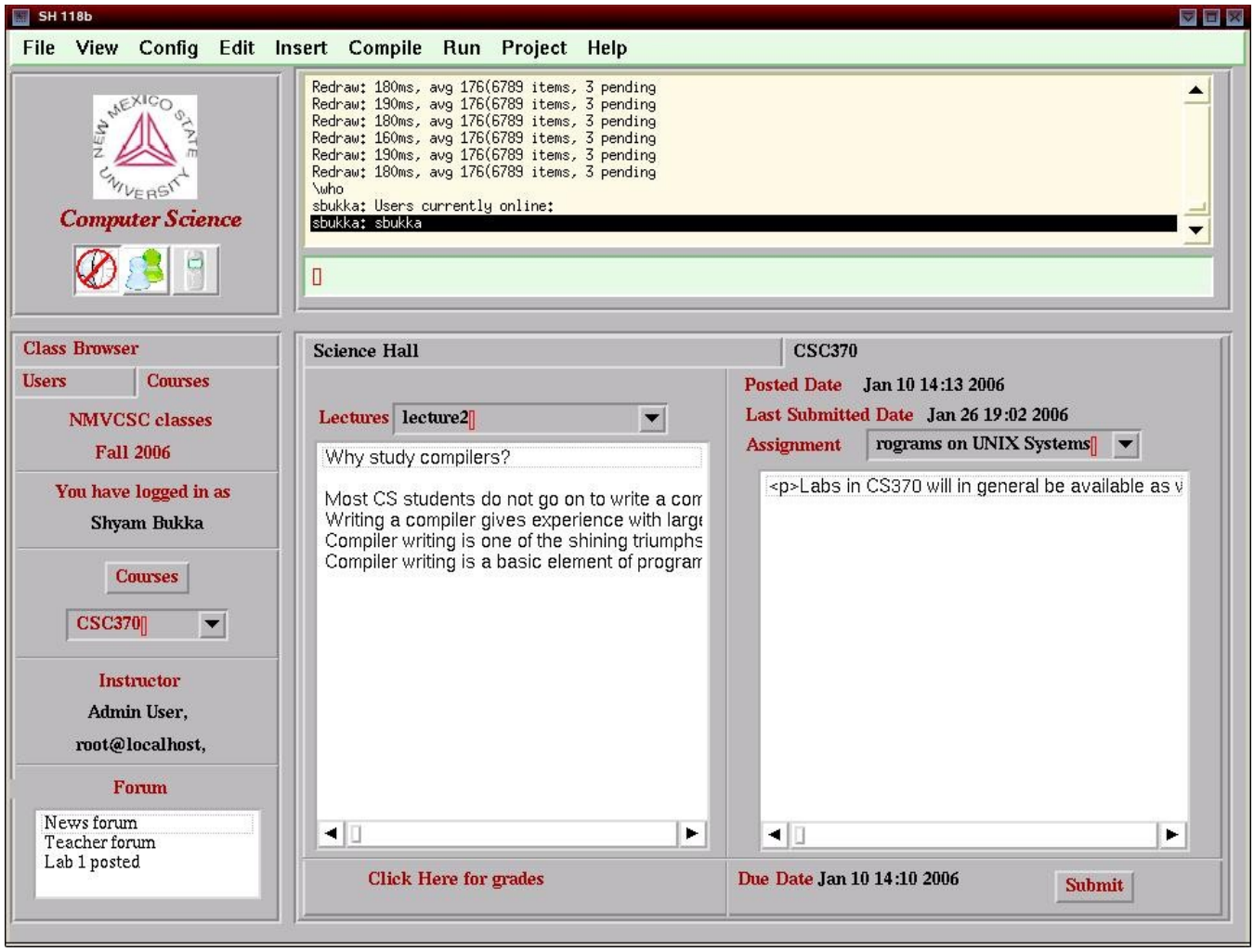


Figure 4: The Assignment Details displayed along with the due time, posted time and the last submitted time

In the above figure, upon selection of an assignment for a particular course, the problem statement is displayed to the user. The due time for the assignment and the posted time of the assignment are also displayed. The last assignment submitted is also displayed to the user.

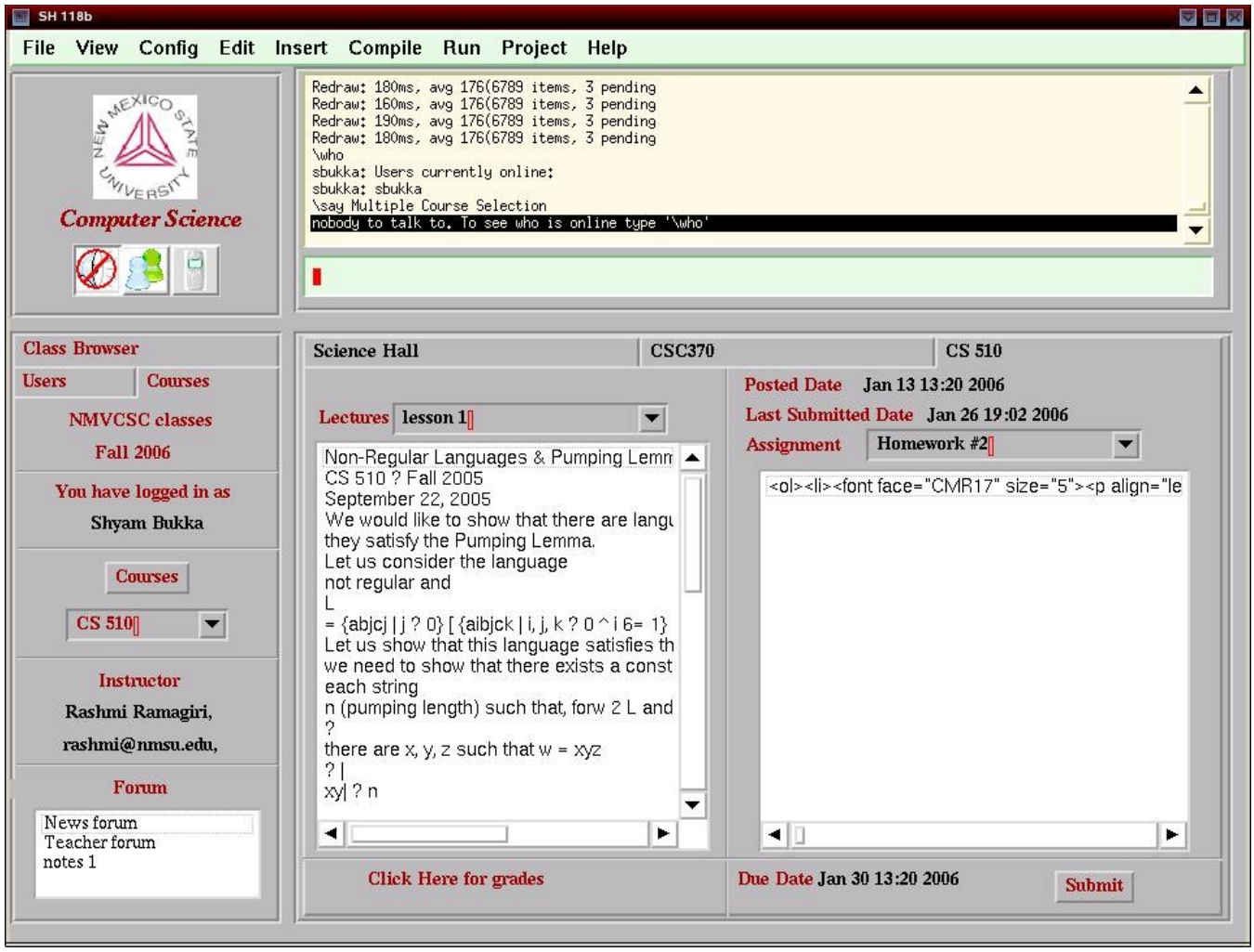


Figure 5: User viewing two courses from the CVE.

If more than one course is selected by the user, a new tab is opened for each course. When a user has to submit an assignment, upon clicking the submit button a file upload dialog box is shown which allows the user to select the file to be uploaded.

8. Related work

Virtual environments like Unicron, Croquet help in creating a collaborative virtual space for the user. These help in teacher-teacher and student-teacher peer communication. Environment such as

Unicron with integrated moodle features are related to the collaborative learning environment which is designed to facilitate the teachers in the field of course management for their students.

Future Learning Environment and Croquet are collaborative virtual environments which are used in the field of education. Future Learning Environment (Fle3) is software for computer supported collaborative learning (CSCL) [Leinonen]. Fle3 is server software for computer supported collaborative learning, designed for group activities supporting creation and development of knowledge artifacts. Knowledge production is the main idea behind CSCL, this can be obtained by making use of advanced computer tools and creating a peer network among the members of the community which includes both teachers and students.

In most of the CSCL applications the knowledge production takes place in a shared working space where students add to the database their knowledge products and carry out progressive discourse interaction. Fle3 can be used just with a single PC computer. Each user may login to the system with their own user name and password and work with the Fle3, rotating the use of the computer. Fle3 users, teachers and students, can use Fle3 with standard web browsers. Fle3 is designed to work with every web browser on every operating system and is also usable with standard web browsers in hand held computers and mobile phones (e.g. Nokia Communicator). Fle3 is easy to localize to different languages and it currently supports twelve languages.

Croquet provides resource sharing and collaboration among users, with the help of open source software with network architecture [Croquet]. It defines a skeleton for delivering constant, scalable and extensible interface to network delivered resources. Croquet allows knowledge sharing, co-creativity and social presence among the users with the help of 2D and 3D interface.

In a 3D architecture, the users can create and publish their own resources. Any number of private or shared worlds can be created by the users and can be made accessible for others by providing spatial portals. The users can also enjoy telepresence with one another. It supports real-time interactions that support a self-organizing, interdisciplinary knowledge-sharing system. Croquet makes use of Open GL-based graphics engine and late binding scripting language to create private or shared authorship of complex spaces and their contents. Croquet large scale networked community allows naïve 3D developers to create shared open-source central repository for storage and retrieval of all created and modified objects. It supports real-time viewing and manipulation of all deliverable information resources across the network. Croquet provides scalable, persistent 3D environment, which ensures that actions and behaviors of an infinite number of networked users are simultaneously apparent to users across the environment.

9. Conclusion and Future Work

With the integration of Moodle features into unicron virtual collaborative environment, it is helpful for a student to view the courses while being present virtually. The interaction between the instructor and the student has been enhanced through the virtual environment by keeping track of all the courses a student is enrolled into and allows the student to use the environment as an internet-webpage for the course site where he/she can view the assignments, lectures and submit the assignments

As the Collaborative Virtual Environment develops, it will influence more students and instructors to use the environment for the purpose of education due to its single consistent and easily accessible interface. Hence there is a need for extending the current environment to one which fully integrates the features provided by the Moodle software and by the commonly used courseware tools.

The chat messages and user login information are currently being logged and kept track of in the virtual environment. Work can be done to include profile pictures in the chat window and allow support for smiles, images, etc. and also in allowing the student to check his grades, enroll for any courses offered and solve quizzes from the virtual environment.

Partial integration of moodle has been done, the modules implemented involved few simple queries like select, insert operations. The performance degradation of unicron with the mysql operations was subtle.

Work can also be done in extending the virtual environment to support most of the features of the Moodle software to make the student's experience better with support to email while using the virtual environment, and by building an online profile including photos and description. Few more features can be added wherein the instructors can enroll/unenroll students manually, include workshops, Quizzes, Choices and Surveys.

Though lot of work has been done from the student side integration of unicron, no or little work has been done from the teacher point of view.

10. References

[Easysoft] Easysoft Limited. (2006). *Linux/UNIX ODBC.* Retrieved December 25, 2005, from http://www.easysoft.com/developer/interfaces/odbc/linux.html

[Leinonen] Leinonen, T., & Kligyte, G. (2002). Future learning environment for collaborative knowledge building and design. *Think Cycle.* Retrieved December 25, 2005 from http://www.uiuh.fi/~tleinone/leinonen_fle3_os.pdf

[Jeffery05] Jeffery, C., Dabholkar, A., Tachtevrenidis, K., Kim, Y. (2005). A Framework for Prototyping Collaborative Virtual Environments. *CRIWG,* Retrieved December 21, 2005 from http://www.cs.nmsu.edu/~jeffery/vcsc/vcsc.pdf

[Jeffery03] Jeffery, C., Mohamed, S., Pereda, R., and Parlett., R. (1993-2003). Programming with Unicon. Retrived December 21, 2005 from http://unicon.sourceforge.net/book/ub.pdf, 2003.

[Balbi02] Jeffery, C., and Balbi, F. (28 June 2002). An ODBC interface for unicon programming language. *Unicon Technical Report #1b*. Retrived December 21, 2005 from http://unicon.org/utr/utr1/utr1.htm

[Croquet] *About Croquet.* (2005) Retrieved December 29, 2005 from http://www.opencroquet.org/

[Guzdial] Guzdial, M. (2005). CaMILE: Collaborative and Multimedia Interactive Learning Environment. Retrieved January 23, 2006, from College of Computing, Georgia Institute of Technology Web site: http://www.cc.gatech.edu/gvu/edtech/CaMILE.html

[Churchill] Elizabeth, F., Churchill., David, S., & Alan, M., (2001). *Collaborative Virtual Environments: Digital Places and Spaces for Interaction.* Springer-Verlang London Ltd
[moodle] *Moodle documentation.* (2006) Retrieved December 18, 2005 from http://docs.moodle.org/