EIJEST

The Egyptian Int. J. of Eng. Sci. and Technology
Vol.14, No. 2 (May 2011)

MOODLE ARCHITECTURE THROUGH REVERSE ENGINEERING*

Ahmed E. Hassan, Mahmoud S. Kandeel**, Aziza S. Asem +, and Mona M. Mowafy+**[1]

**Faculty of Engineering , Mansoura University, Egypt
+ Faculty of Computer Science & Information Systems , Mansoura University, Egypt

ABSTRACT

The course management system (CMS) is a basic software piece of any educational institution. Web enabled CMS is a tool to handle large number of students institutions. There are many open sources CMS, they are not having software engineering based documentations. This leads to a lot of problems in its maintenance and operation. These systems do not consider the locality and policy of institutions. All open source CMS do not support quality assurance concepts in education. Authors of that research proposed a CMS based on reverse engineering of the open CMS "Moodle system". Authors applied their reverse engineering approach (Ref. [1]) on moodle to come up with new CMS. The proposed CMS is more maintainable, extendable, and technically updatable. Moodle CMS is analyzed to identify the proposed system's components and their interrelationships. The functionality of the proposed CMS could be improved to be suitable for university with different faculties and departments. New modules for the proposed CMS such as annual course report could be easy to plug to the proposed CMS.

KEY WORDS: Reverse engineering, Course management system, UML analysis

ARCHITECTURE MOODLE PAR INGENIERIE INVERSE ARCHITECTURE MOODLE PAR INGENIERIE INVERSE

RÉSUMÉ

Le système de gestion de cours (CMS) est un morceau du logiciel de base de tout établissement d'enseignement. Web activé CMS est un outil pour gérer grand nombre d'institutions étudiants. Il ya beaucoup de sources ouvertes CMS, ils ne sont pas d'avoir les documentations d'ingénierie logicielle basée. Cela conduit à beaucoup de problèmes dans son entretien et l'exploitation. Ces systèmes ne considèrent pas la localité et de la politique des institutions. Tous les CMS open source ne soutiennent pas des concepts d'assurance qualité dans l'enseignement. Les auteurs de cette recherche a proposé un CMS basé sur le reverse engineering du «système de Moodle" CMS open. Auteurs appliqué leur approche d'ingénierie inverse (Réf. 1) sur Moodle à venir avec de nouvelles CMS. Le CMS proposé est plus maintenable, extensible, et techniquement actualisables. Moodle CMS est analysé afin d'identifier les composants du système proposé et leurs interrelations. La fonctionnalité de la CMS proposé pourrait être améliorée pour être adapté à l'université avec différents départements et facultés. De nouveaux modules pour le projet de CMS tels que le rapport annuel pourrait être bien sûr facile à brancher à la proposition de la CMS.

MOTS CLÉS: REVERSE ENGINEERING, SYSTEME DE GESTION DE COURS, L'ANALYSE UML.

* Received: 2/2/2011, accepted: 4/5/2011 (Technical Report)
(1) Contact author (0106735534)

1- INTRODUCTION

1.1 Overview

Course Management System is web application. It follows client/ server technique and is accessed by client using a web browser. The system server is probably located in The university or the department, but it can be anywhere in the world.

CMS as defined in (Ref. [2]) is a collection of software tools providing an online environment for course interactions. A CMS typically includes a variety of online tools and environments, such as:

- An area for institution posting of class materials such as course syllabus and handouts
- An area for student posting of papers and other assignments
- A gradebook where institution can record grades and each student can view his or her grades
- An integrated email tool allowing participants to send announcement email messages to the entire class or to a subset of the entire class
- A chat tool allowing synchronous communication among class participants
- A threaded discussion board allowing asynchronous communication among participants

In addition, a CMS is typically integrated with other databases in the university so that students enrolled in a particular course are automatically registered in the CMS as participants in that course.
According to (Ref. [3]), once institution start to use a CMS, their use of the technology tends to grow. Many institution spoke of how using the CMS allowed them to see new ways that they could use it in their classes or ways they might use it in a different type of class. Other institution spoke of how they learned about new uses and applications of a CMS from discussions with colleagues (especially with those from other disciplines) or in training sessions. When they include practical examples of the use of the software, these influences and sessions seem especially effective in promoting the increased use of course management systems
In the other side some institution whose use of course management systems decreased over time related that the technology was time consuming, inflexible, and difficult to use. They had few complaints about the time spent restructuring and redesigning courses in order to incorporate a CMS in their instruction, but they resented the time required to load and reload course materials. It is this time expenditure that frequently contributed to some institution’s reducing their CMS use. Other institution found course management systems inflexible and overly-structured, while a significant number of others complained that most course management systems could not easily handle mathematical and scientific notation. Some institution found course management systems too difficult to use, and a greater number limited their use of a CMS because of problems that students reported. This emerged as a significant problem affecting the rate, level, and success of CMS use. All institution said their use of course management systems would grow if the software were easier to use and if training—for themselves and their students—were more available.

1.2 CMS Advantages

Decentralized maintenance. Typically based on a common web browser. Edit anywhere, anytime. Bottlenecks removed.
Designed with non-technical content editors in mind. People with average knowledge of word processing can create the content easily. No HTML skills required.
Configurable access restrictions. Users are assigned roles and permissions that prevent them from touching content in which they are not authorized to change.
Consistency of design is preserved. Because content is stored separate from design, the content from all authors is presented with the same, consistent design.
Navigation is automatically generated. Menus are typically generated automatically based on the database content and links will not point to nonexistent pages.
Content is stored in a database. Central storage means that content can be reused in many places on the website and

formatted for multiple devices (web browser, mobile phone/WAP, PDA, printer).

Dynamic content. Extensions like forums, polls, shopping carts, search engines, news management are typically drop-in modules. A good CMS also allows for truly user defined extensions.

Daily updates. You do not need to involve web designers or programmers for every little modification - you are in control of your website.

Cooperation. Encourages faster updates, enforces accountability for content editors via log files and promotes cooperation between authors.

Content scheduling. Content publication can often be time-controlled; hidden for previews; or require a user login with password.

2- EVALUATION OF MOODLE AS CMS

According to (Ref. [4]) open Source software differs from normal commercial software only in that the human readable source code is available to the user so that they can inspect and modify it. Many Open Source projects also have computer-readable executable files available for easy installation. In contrast, most commercial companies will carefully guard the source code for their products, treating the source as a trade secret. The user receives only a computer-readable executable and may not look at or modify the internal code.

The development philosophy of Open Source is different than for closed source products. Rather than having a dedicated team of developers working at a central company, carefully managed to produce a saleable product, the Open Source ideal is that thousands of users will download and inspect the code as it is released. The users themselves will find the bugs and fix them, as well as adding features that they feel are necessary. These small fixes should occur quickly, enabling rapid and steady growth in a software package as a whole: "release early, release often". The Internet allows users distributed around the globe to collaborate almost as easily as if they were in the same room.

Based on(Ref. [5]), open source systems has many advantages .Such as the upgrading and updating are free, just like the initial software download. Institutions can install open source products without worrying that the costs will increase. In addition, these low costs mean that educators and technology staff can experiment with different software to find the software that is the best fit for district needs; with commercial software Institutions must often rely on demonstrations in a limited time frame or with use restrictions.

Also, open source software allows the technology staff the freedom "*to customize and 'get under the hood*'" (Ref. [6]). With the right training, technology staff can 'tweak' products to meet the needs of the users. This freedom can be expanded into learning opportunities for students as well. Open source product use and development often involves problem solving and collaboration. If students are involved in the selection, installation or maintenance of open source products, they are being exposed to valuable learning opportunities.

Open source software technical, tech support is free and available at all hours through the community of developers and users surrounding the open source project. This community of developers also ensures continued growth and development of features that are meaningful to the end user. Programmers on open source products work to create features that are useful for themselves or their end users; money is not the motivation as it might be in commercial software. The development community also increases the likelihood that end users will not find themselves faced with a company who has no longer decided to support, improve, or sell a product. In most open source development communities there are always new members joining to work on the project.

Moodle is an Open Source Course Management System (CMS), also known as a Learning Management System (LMS) (Ref. [7]) or a Virtual Learning Environment (VLE) (Ref. [8]). It has become very popular among educators around the world as a tool for creating online dynamic web sites for their students. To work, it needs to be installed on a web server somewhere, either on one of your own computers or one at a web hosting company.

The focus of the Moodle project is always on giving educators the best tools to manage and promote learning, but there are many ways to use Moodle:

- Moodle has features that allow it to scale to very large deployments and hundreds of thousands of students, yet it can also be used for a primary school or an education hobbyist.
- Many institutions use it as their platform to conduct fully online courses, while some use it simply to augment face-to-face courses (known as blended learning).
- Many of our users love to use the activity modules (such as forums, databases and wikis) to build richly collaborative communities of learning around their subject matter (in the social constructionist tradition), while others prefer to use Moodle as a way to deliver content to students (such as standard SCORM packages) and assess learning using assignments or quizzes.

There are many of Open source CMS but the moodle is considered as the most full featured system. In this research the moodle features is considered as a reference of comparison moodle with the features of other open sources CMS. This comparison showed that moodle features doesn't completely exist in any open source CMS. There are some features which are found in most of systems because they considered as foundation of any CMS, while there are features that exist in some system and don't exit in others.
The features of the moodle system are compared with the features of the open sources CMS as in (Ref. [9-31]).The following table shows the name of each feature form moodle features and the percentage means the number of CMS systems that has this feature divided by the total number of the CMS systems under comparison (Ref. [9-31]).

The Following chart (Fig. (1)) represents the percentage for each feature from moodle features according to the other system

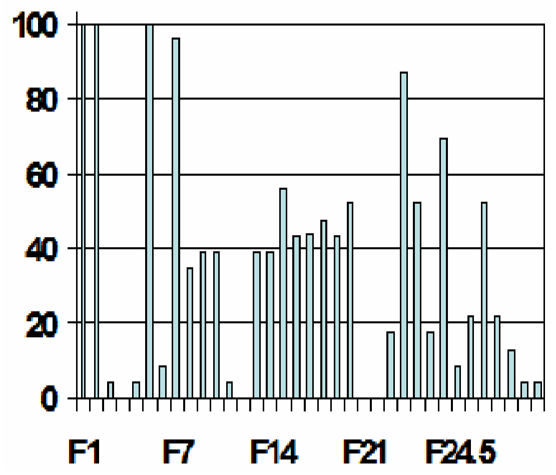


Fig.(1):Visual Percentage of Features as Calculated in Table (1).

This chart reflects the previous table which show visually how the features found in the system assuming that features are 100 % in moodle.
The diagrams shows there are some features such as F1, F2, F6.1 are found in all systems because they considered as foundation of any CMS, while there are features such as F4, F12, F21, F22 that don't exist the other systems. The other Features exits in some systems and don't exit in others. So the moodle has features that not completely found in the other systems. These features make the system more usable, comfortable for the course management process .That is why that made the author to choose the moodle system

3- REVERSE ENGINEERING FOR MOODLE SYSTEM

3.1 Overview

The moodle course management system was reverse engineered using the approach described in details as a process for reverse engineering in the previous paper [1].

As shown in Fig. (2), the approach consists of three phases every one has sub phases. Applying all that phases and sub phases in details for Moodle is explained in the remain of this section.

Table (1):- Features Names and Their Availability Ratios

Symbol	Feature Name	Ratio
F1	Site managed by admin	100%
F2	Site can be modified by site administration block	100%
F3	Plug-in themes	4.3%
F4	Plug-in activity modules	0%
F5	Plug-in languages packs	4.3%
F6	User authentication mechanisms	
F6.1	Standard mail method	100%
F6.2	LDAP method	8.7%
F7	Building an online edit profile	95.7%
F8	Teachers can add enrollment key to their courses	34.8%
F9	Enrolling students manually	39.1%
F10	Unenrolling students manually	39.1%
F11	Enrollment plugins	4.3%
12	Meta courses	0%
F13	Course Formats settings	39.1%
F14	Force theme for any courses	39.1%
F15	Groups	56.5%
F16	Recent changes can be displayed on the course Homepage	43.5%
F17	HTML editor	34.8%
F18	Gradebook	47.8
F19	Full user logging and tracking reports for each student	34.5
F20	Mail integration	52.2
F21	Custom scales	0%
F22	Backup for courses and site	0%
F23	Activities and resources can be imported another courses	17.4%
F24	Flexible array of courses activities	
F24.1	Assignment Module	86.96
F24.2	Chat Module	52.2
F24.3	Choice Module	17.4
F24.4	Forum Module	69.6
F24.5	Glossary Module	8.7
F24.6	Lesson Module	21.7
F24.7	Quiz Module	52.2%
F24.8	Resource Module	21.7%
F24.9	Survey Module	13.04%
F24.10	Wiki Module	4.3%
F24.11	Workshop Module	4.3%

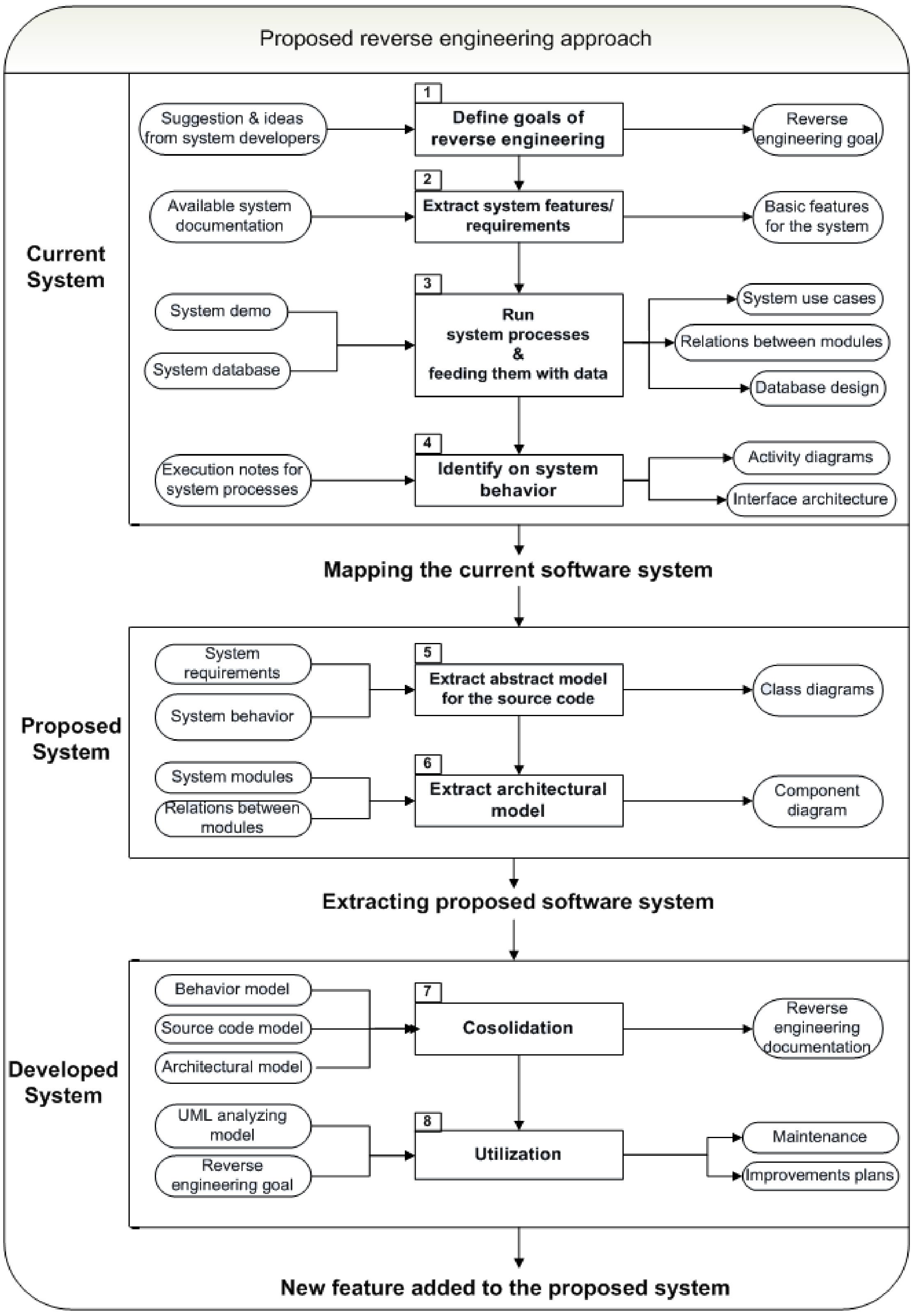

Proposed reverse engineering approach
Current System
Suggestion & ideas from system developers
1 Define goals of reverse engineering
Reverse engineering goal
Available system documentation
2 Extract system features/ requirements
Basic features for the system
System demo
System database
3 Run system processes & feeding them with data
System use cases
Relations between modules
Database design
Execution notes for system processes
4 Identify on system behavior
Activity diagrams
Interface architecture
Mapping the current software system
Proposed System
System requirements
System behavior
5 Extract abstract model for the source code
Class diagrams
System modules
Relations between modules
6 Extract architectural model
Component diagram
Extracting proposed software system
Developed System
Behavior model
Source code model
Architectural model
7 Cosolidation
Reverse engineering documentation
UML analyzing model
Reverse engineering goal
8 Utilization
Maintenance
Improvements plans
New feature added to the proposed system

Fig. (2): Reverse Engineering Approach

3.2 Phase 1 (Current System):

This phase take the suggestions and ideas from system developers , available documentations for the system , system demo and execution notes for system processes as inputs to produce reverse engineering process goals , basic features for the system , system use cases and activity diagrams as output for this phase. This phase works as an analyzing stage for the current system and mapping it. This phase consists of four sub phases as shown in the following details.

Define goals of reverse engineering process for moodle system:

- Institution whom use or develop new features in moodle CMS faces the problem of non-availability of analysis and design documentation, so they need analyze the system using the reverse engineering technique to identify the system's components and their interrelationships, create representations of the system in another form or a higher level of abstraction and create the physical representation of that system**.** Institutions could develop their CMS based on reverse engineered moodle.
- Moodle in not suitable for universities with different faculties and departments. To fit with these kinds of institutions, Moodle has to modify in its structure. After reverse engineering of Moodle, it’s easy to rebuild a new CMS which fit the needs of new type of institutions.

Extract system features \ requirements

Using the available documents for the system we could to extract the basic features for the system

Moodle's overall design:

- Promotes a social constructionist pedagogy (collaboration, activities, critical reflection, etc)
- Suitable for 100% online classes as well as supplementing face-to-face learning
- Simple, lightweight, efficient, compatible, low-tech browser interface
- Easy to install on almost any platform that supports PHP. Requires only one database (and can share it).
- Full database abstraction supports all major brands of database (except for initial table definition)
- Course listing shows descriptions for every course on the server, including accessibility to guests.
- Courses can be categorized and searched - one Moodle site can support thousands of courses
- Emphasis on strong security throughout. Forms are all checked, data validated, cookies encrypted etc
- Most text entry areas (resources, forum postings etc) can be edited using an embedded WYSIWYG HTML editor

Site management

- Site is managed by administrator user
- Site is defined during setup. Defaults can be edited during setup or globally accepted
- Site can be modified by a robust Site administration block.
- Plug-in "themes" allow the administer-ator to customize the site colors, fonts, layout etc to suit local needs
- Plug-in activity modules can be added to existing Moodle installations
- Plug-in language packs allow full localization to any language. These can be edited using a built-in web-based editor. Currently there are language packs for over 70 languages.
- The code is clearly-written PHP under a GPL license - easy to modify to suit your needs

User management

- Overview
- Enrolment
- Roles

Course management

- Overview
- Assignment Module
- Chat Module
- Forum Module
- Glossary Module
- Lesson Module
- Quiz Module
- Resource Module
- Survey Module
- Wiki Module
- Workshop Module

Running system processes and feeding them with data

Running system processes and feeding them with data is the most important step in the proposed approach. To feed the processes with data, it should be important to study how the system's data is related. The relation between data is represented as the ER diagrams for the system Database. The following diagram (Fig. (3)) depicts as example the ER diagram for the courses module in the moodle system

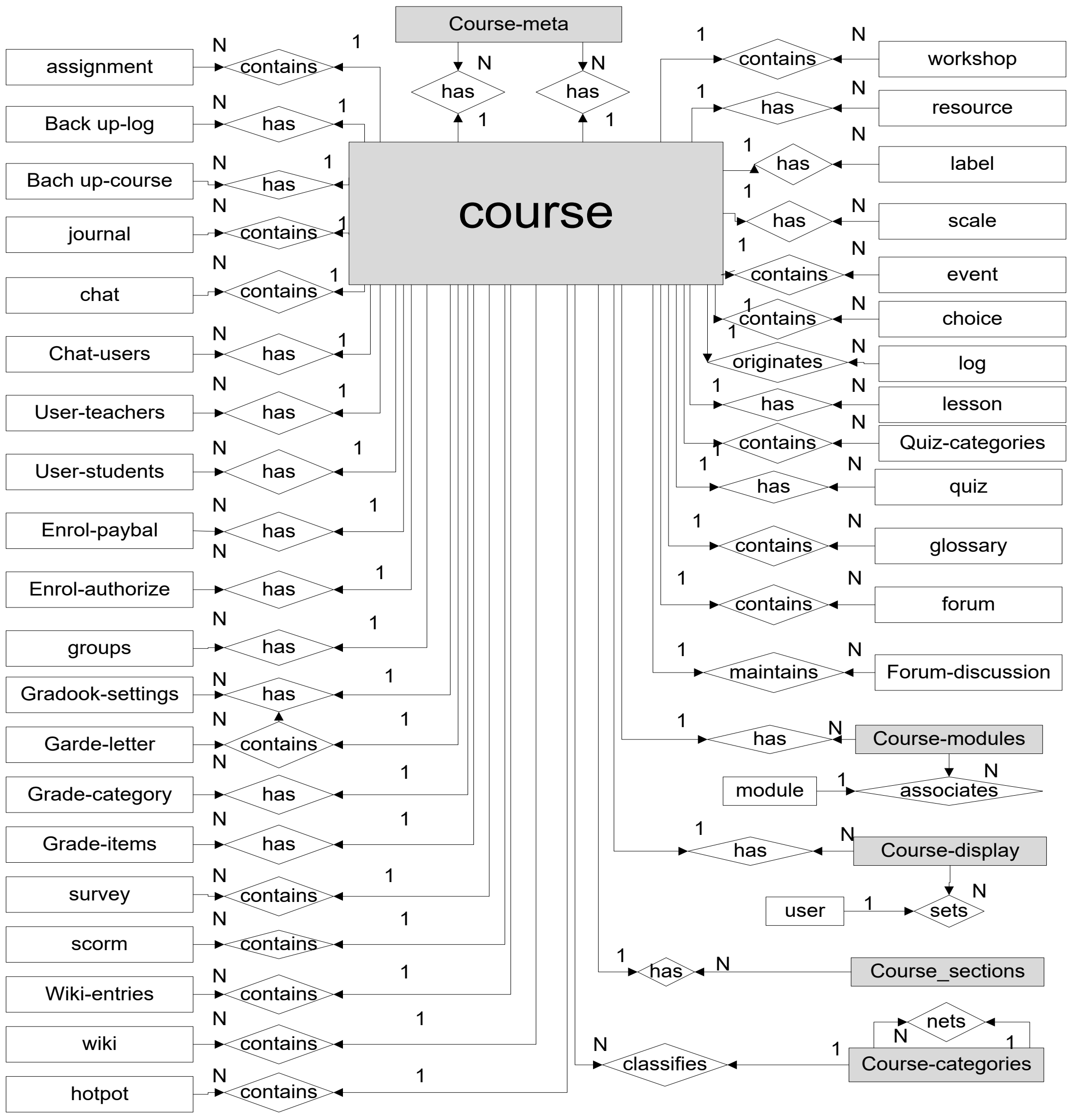


Fig. (3): ER Diagram for Course Module

With understanding how the data is related and how each module related with the other we could run the system processes and feeding them with the data to produce the system use cases diagrams. The following diagram shows the top level for the system use cases (Fig. (4)).

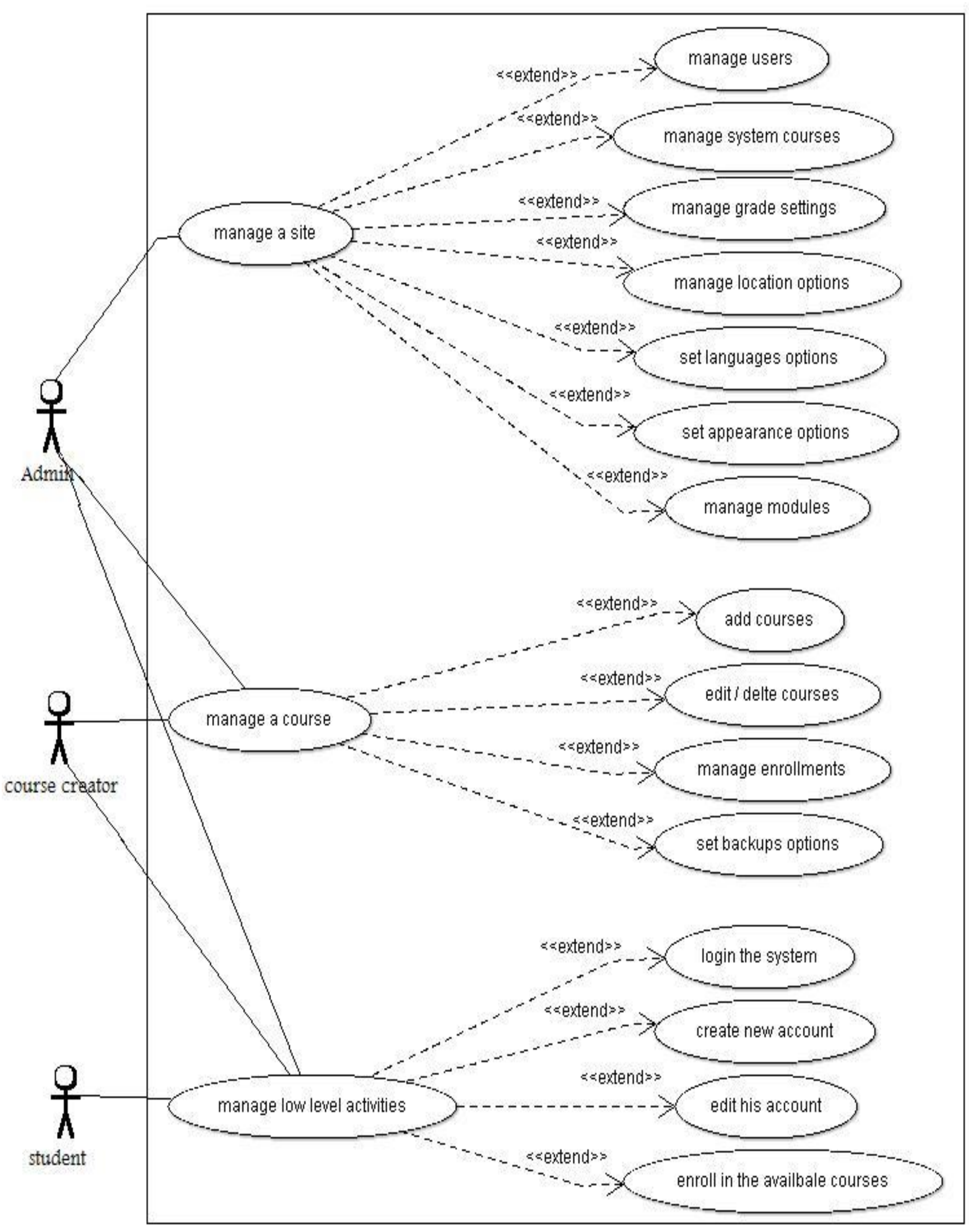


Fig. (4): Top Level of Moodle Use Cases

3.3 Phase 2 (Proposed System):

This phase take system requirements, system behavior, system modules and relations between them as inputs to produce the class diagrams and component diagram as output for this phase**.** This phase works as designing stage for the system but from the point view of the reverse engineer as he views the system to put the system in new form. This phase consists of two sub phases as shown in the following details.

Identify on system behavior

Depending on the notes that are observed from the previous phase we could produce the activity diagrams for system processes. The following diagram (Fig. (5)) depicts the activity diagram for the whole system

The following diagram (Fig. (6)) depicts the activity diagram for adding or editing system courses

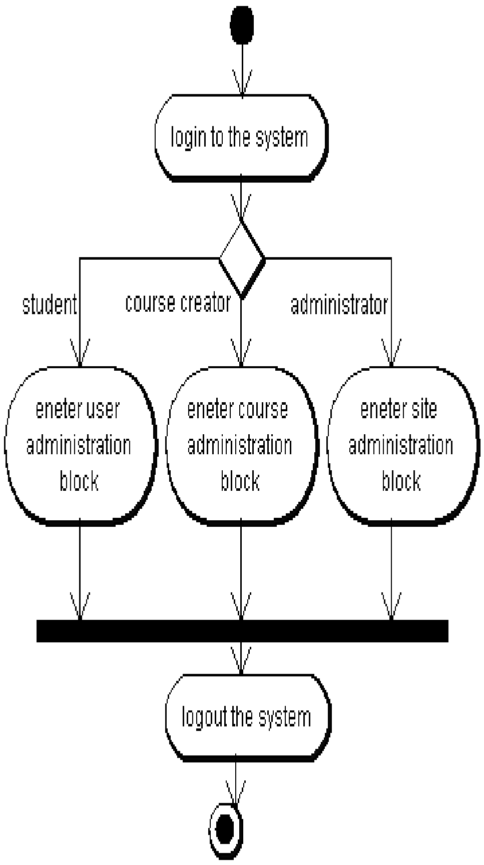

login to the system
student
course creator
administrator
eneter user administration block
eneter course administration block
eneter site administration block
logout the system

Fig. (5): Activity Diagram for Logging Process

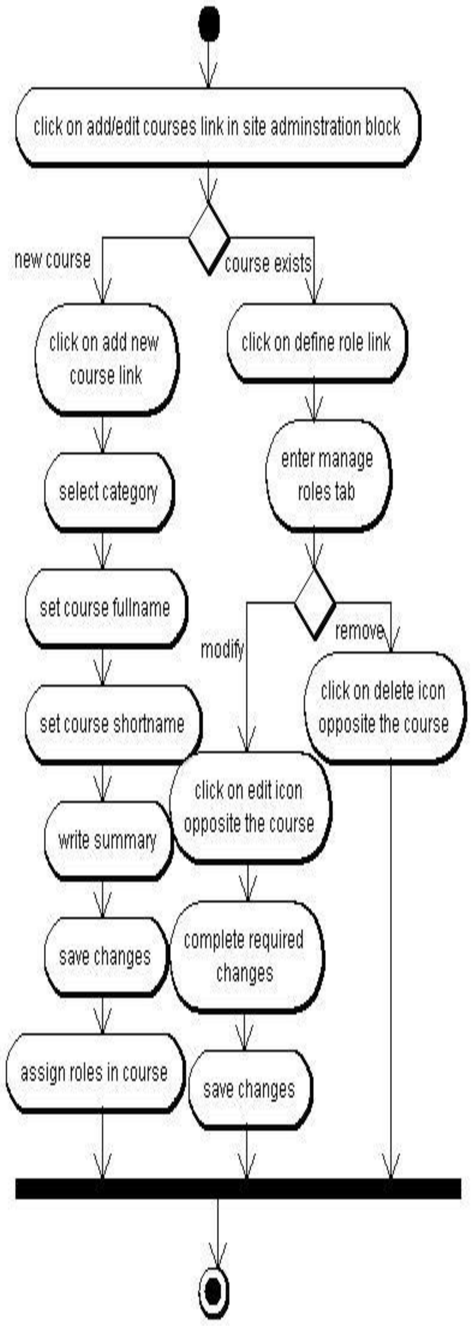

click on add/edit courses link in site adminstration block
new course
course exists
click on add new course link
click on define role link
select category
enter manage roles tab
set course fullname
modify
remove
set course shortname
click on delete icon opposite the course
click on edit icon opposite the course
write summary
save changes
complete required changes
assign roles in course
save changes

Fig. (6): Activity Diagram for Adding or Editing New Course

The following diagram (Fig. (7)) depicts the Cenario diagram for the process of adding New course to the system

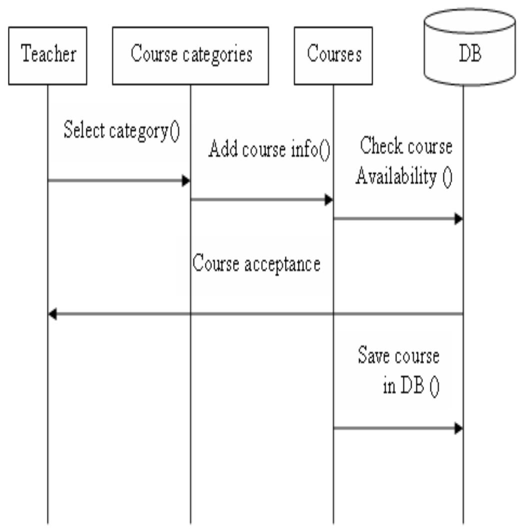


Fig. (7): Scenario Diagram for Adding New Course

Now, it is more easier to identify on the interface architecture for the system, the following diagrams (Figs. (8), (9), and (10)) represent the interface structure for the section of editing courses in the system.

Extract abstract model from the source code
With defining system requirements and system behavior model, we could put imaging for the source code model through the class diagrams for the system processes. The following diagram (Fig. (11)) represents as example the class diagram for the course module in the moodle system.

Extract architectural model
With the understanding gained out of performing the above steps and reading the developer's documentation, the architectural model will be easier to be designed as depicted in the following diagram (Fig. (12)) that represents the component diagram for the moodle system.

3.4 Phase 3 (Developed System):

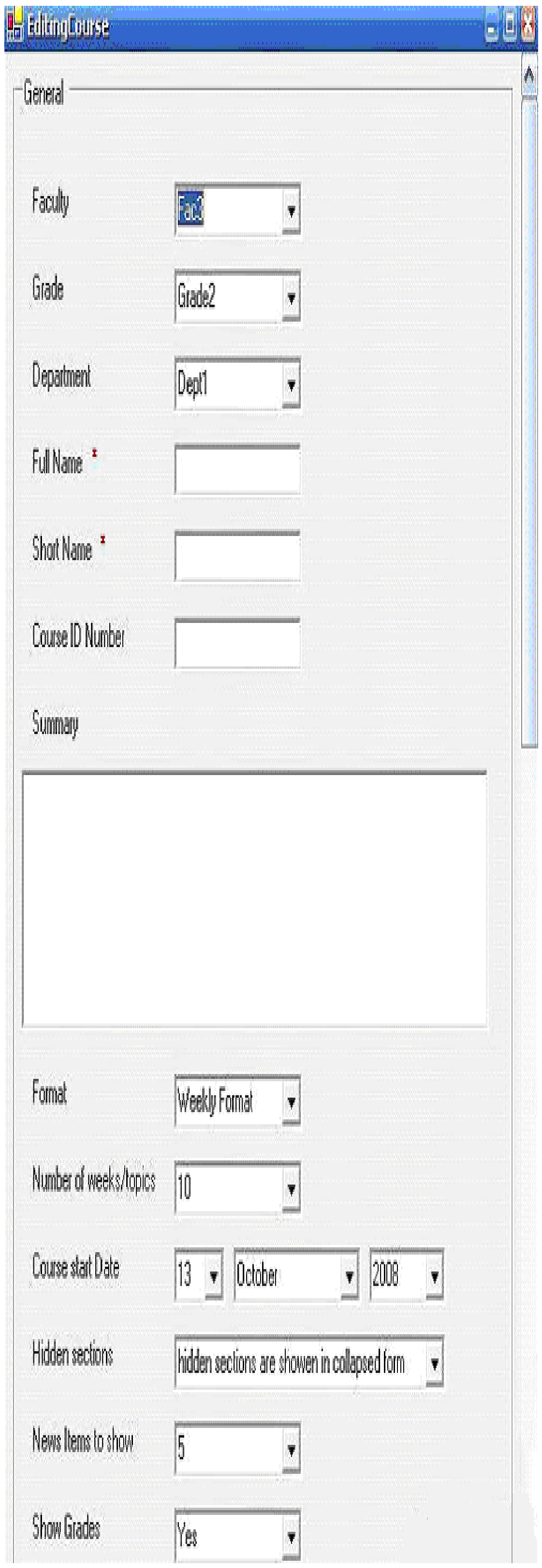
This phase takes the behavior model , source code model , architectural model , UML analyzing model and reverse engineering goal as inputs to produce the reverse engineering documentation and maintenance and improvements plans as outputs for this phase . This phase works as the resulting stage for the reverse engineering process to produce the new system with the proposed modifications. This phase consists of two sub phases as shown in the following details.

Consolidation
Once the various models were built, the consolidation phase took hold. The models were studied again in light of the goals
specified during the beginning of the project. Complex models that could potentially be unclear than the source code were removed from the model. Some models were enhanced when they were attached to another view of the system.

Utilization
The end result of this project was one coherent UML model that correlates all the knowledge gained. Analyzing the UML model generated and using it for future maintenance and development would determine the utility of this reverse engineering effort

4- CONCULSION

At that research a CMS system is proposed which is based on a reverse engineering approach. The proposed CMS used a moodle open source CMS to doing it reengineering process.
For institution, to cope with CMS era, it has to follow one of the following scenarios:
First: Institutions uses one of the available open source CMS. This approach leads to limitation with the current system capabilities so the institution must adapt its needs to be suitable with the system capabilities. Second: Institutions could build its own CMS system from scratch. Building CMS system from scratch is time and cost consuming. Third: Institutions reverses engineer one of the available CMS and build new one. .In this scenario the risk of losing critical business knowledge, which may be embedded in a legacy system or producing a system that doesn't meet its users' real need, is drastically reduced.
Authors of that research proposed a CMS system following up the third scenario.
At that research a CMS system is proposed which is based on a reverse engineering approach. The proposed CMS used a reengineering process for the moodle open source CMS.

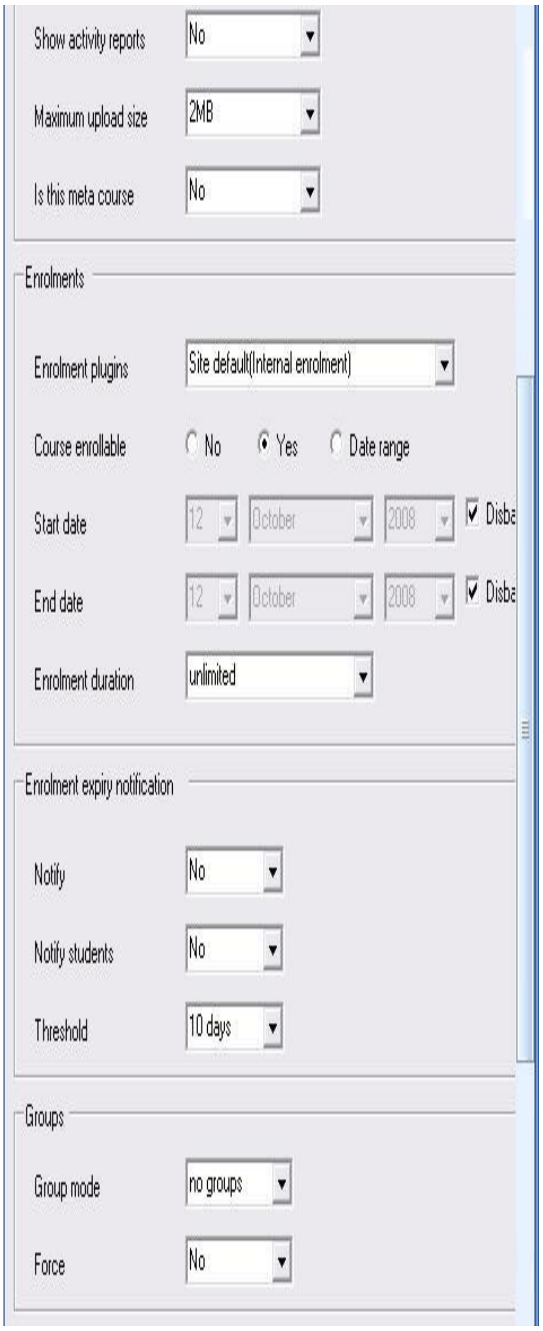


EditingCourse

General

Faculty	Fac3
Grade	Grade2
Department	Dept1
Full Name *	
Short Name *	
Course ID Number	
Summary	
Format	Weekly Format
Number of weeks/topics	10
Course start Date	13 October 2008
Hidden sections	hidden sections are showen in collapsed form
News Items to show	5
Show Grades	Yes

Fig. (8): Editing Course Page1



Show activity reports	No
Maximum upload size	2MB
Is this meta course	No

Enrolments

Enrolment plugins	Site default(Internal enrolment)
Course enrollable	No [✓ Yes] Date range
Start date	12 October 2008 ✓ Disa
End date	12 October 2008 ✓ Disa
Enrolment duration	unlimited

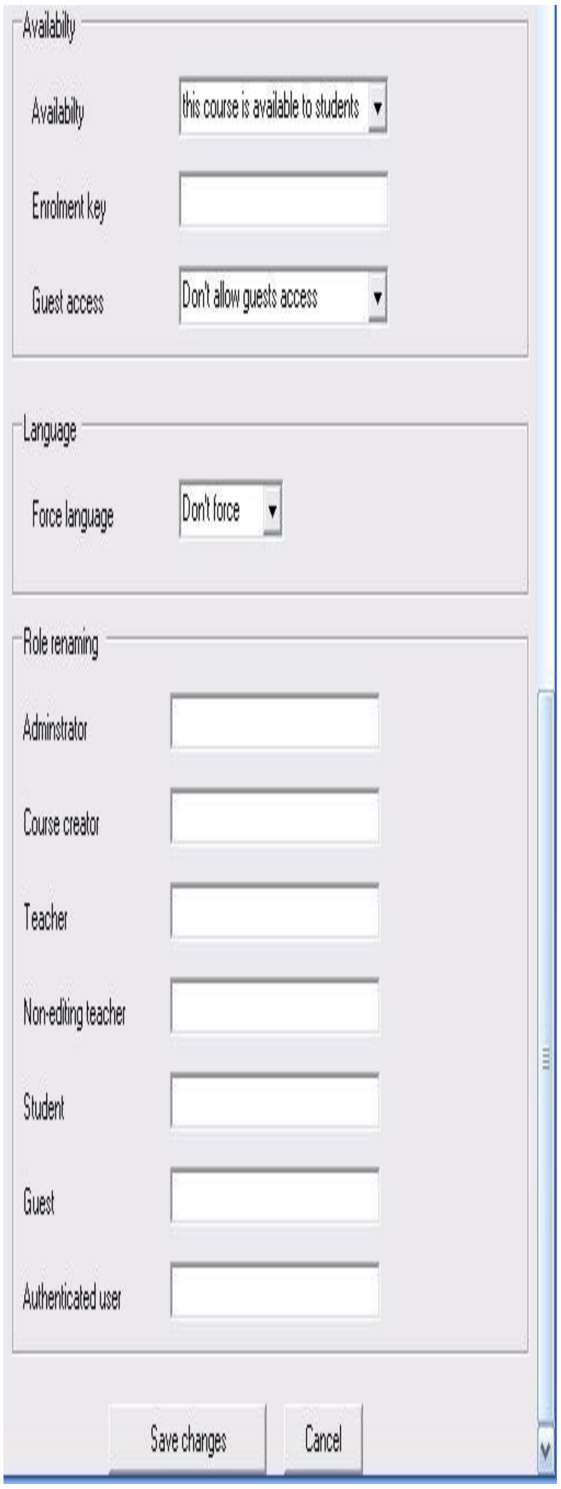
Enrolment expiry notification

Notify	No
Notify students	No
Threshold	10 days

Groups

Group mode	no groups
Force	No

Fig. (9): Editing Course Page2



Availabilty

Availabilty	this course is available to students
Enrolment key	
Guest access	Don't allow guests access

Language

Force language	Don't force

Role renaming

Adminstrator	
Course creator	
Teacher	
Non-editing teacher	
Student	
Guest	
Authenticated user	

Figure (10): Editing Course Page3

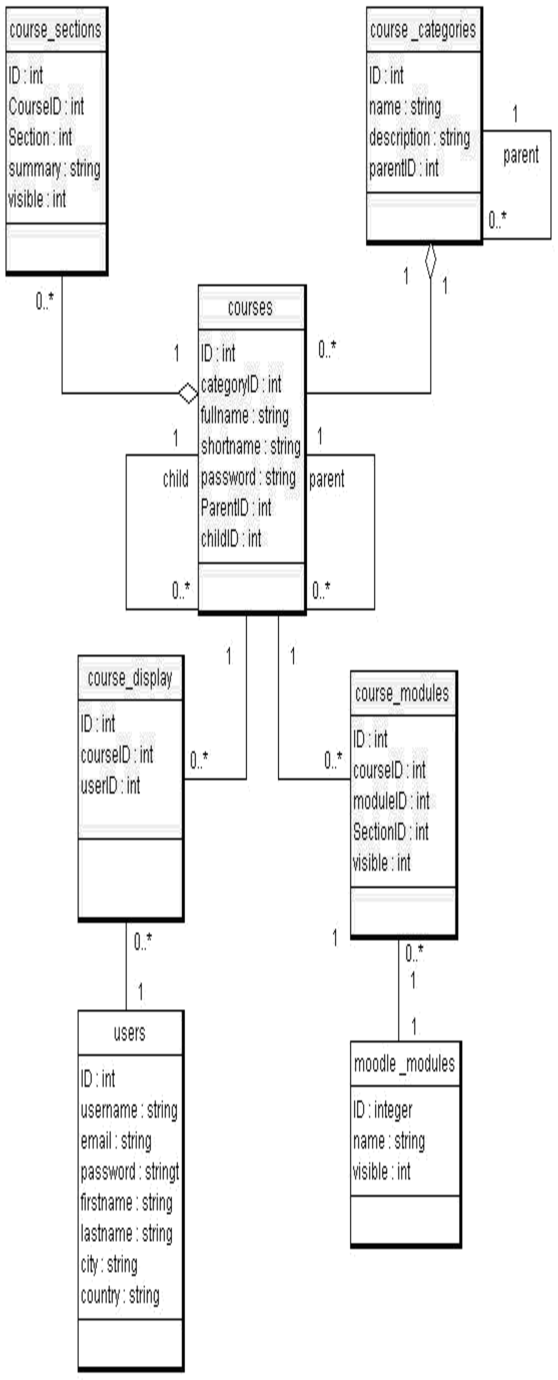


Fig. (11): Class Diagram for Course Module

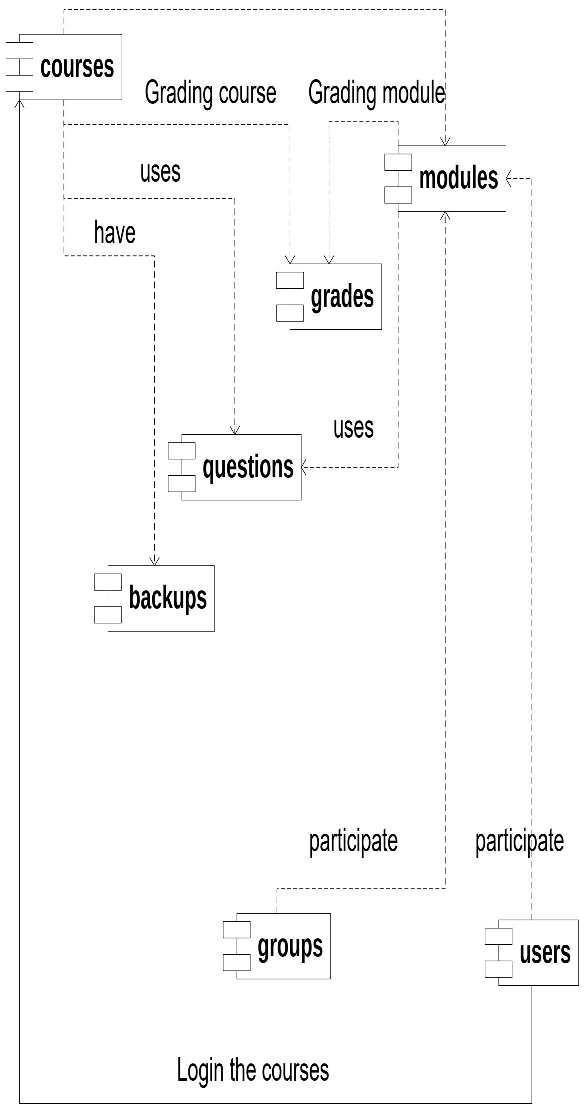


Fig. (12): Component Diagram for the Moodle System

The moodle open CMS system has features that not completely found in the other open systems. These features make the system more usable, comfortable for the course management process .

That is why made the author to choose the moodle system to be reverse engineered using the proposed reverse engineering approach.
All phases of the reverse engineering are applied and all UML models are developed. Regarding space on time not all UML models shown in that paper. In the future the implementation of the proposed CMS could be done.

In the future work based on quality assurance approaches many modules could be added which make the system coped with the advanced polices needed for a future CMS systems. Authors works on a modules like course report module and Intended Learning Outcomes modules.

REFERENCES

1. Mahmud kandel, Ahmed E. hassan, Aziza S. Asem, Mona M.mowafy, " Reverse Engineering Approach For Web Apllications ", International Journal of Software Engineering and Knowledge Engineering (IJSEKE) (on the press)

2. http://cft.vanderbilt.edu/teaching-guides /technology/course-management-systems/, (August 2010)

3. Glenda Morgan, Key Findings. Faculty Use of Course Management Systems, ECAR Key Findings. May 2003.

4. Eric Rémy , Open Source Course Management Systems: a Case Study , Proceedings of the 2005 ASCUE Conference, www.ascue.org June 12-16, 2005, Myrtle Beach, South Carolina

5. Michelle Moore , Moodle Open Source Course Management System : A Free Alternative to Blackboard , Masters Project in Instructional Design and Technology, December 5, 2003

6. Nelson, P. & Bucknell, D. (2002, Fall). The free software revolution. Scholastic Administrator [Online]. Retrieved October 2003 from

7. Saiid Ganjalizadeh , overview of open source learning management systems , EDUCAUSE Evolving Technologies Committee , September 15, 2006
8. P. Dillenbourg, "Learning in the new millennium: Building new education strategies for schools", Workshop on virtual learning environments, EUN conference, 2000

9.http://web.mit.edu/dekane/www/LRN/Documentation_HTML/Features.htm (1/1/2008)

10.http://opensourcecms.com/index.php?option=com_content&task=view&id= 277&Itemid=1 1/1/2008

11.http://klaatu.pc.athabascau.ca/cgi-bin/ b7/main.pl?rid=32 (2/1/2008)

12.http://bodington.org/choose.php (30/11/2010)

13.http://www.hmc.edu/about/administrativeoffices/cis1/docs1/central1/web1/ sakai.html 30/11/2010

14.http://www.tc.columbia.edu/cis/help/ about2.htm (30/11/2010)

15.http://edukalibre.org/documentation/ moodle_vs_claroline.ps (30/11/2010)

16.http://en.wikipedia.org/wiki/CourseWork_ Course_Management_System (1/12/2010)

17.http://physik.uni-graz.at/~cbl/electure/ (1/12/2010)

18.http://www.velocedge.com/CADEnew/ Products/eTutor/features.htm (1/12/2010)

19.http://en.wikipedia.org/wiki/Fle3 (1/12/2010)

20.http://www.ilias.de/docu/goto.php?target =lm_392&client_id=docu (1/12/2010)

21.http://www.jonesstandard.org/index.php?module=ContentExpress&func= display &ceid=7 (8/1/2008)

22.http://learnloop.sourceforge.net /download.php (1/12/2010)

23.http://cbdd.wsu.edu/edev/Kenet_ToT/ Unit2/KEWLFunctions.htm (9/1/2008)

24.http://www.edutools.info/compare.jsp? pj=8&i=345 (1/12/2010)

25.http://www.edutools.info/compare.jsp? pj=8&i=307 (1/12/2010)

26.http://www.edutools.info/compare.jsp? pj=8&i=349 (1/12/2010)

27.http://www.edutools.info/compare.jsp? pj=8&i=352 (1/12/2010)

28.https://segue.middlebury.edu/index.php?&site=segue&section=5736&page= 23716& action=site (1/12/2010)

29.http://www.campussource.de/org/ software /uop (1/12/2010)

30.http://whiteboard.sourceforge.net/demo /docs/ (1/12/2010)

31.http://webwork.math.rochester.edu/docs/ (8/1/2008)