Problem Based Learning Tool as a Plug-in for Moodle

A Thesis

Submitted in partial fulfillment of the requirements
for the degree of

Master of Technology

by

Souman Mandal

Roll No: 09305066

under the guidance of

Prof. Sridhar Iyer

Department of Computer Science and Engineering
Indian Institute of Technology, Bombay
Mumbai
June, 2011

Dissertation Approval Certificate

Department of Computer Science and Engineering

Indian Institute of Technology, Bombay

The dissertation entitled **"Problem Based Learning Tool as a Plug-in for Moodle"**, submitted by **Souman Mandal** (Roll No: **09305066**) is approved for the degree of **Master of Technology** in **Computer Science and Engineering** from **Indian Institute of Technology, Bombay**.

Prof. Sridhar Iyer
CSE, IIT Bombay
Supervisor

Prof. Sahana Murthy
CDEEP, IIT Bombay
Internal Examiner

Prof. Vijay Raisinghani
HoD, NMIMS
External Examiner

Prof. Girish Saraph
EE, IIT Bombay
Chairperson

Place: IIT Bombay, Mumbai
Date: 28th **June, 2011**

Declaration

I declare that this written submission represents my ideas in my own words and where others ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Souman Mandal
(09305066)

Date: 28th **June, 2011**

Acknowledgement

I take this opportunity to express my sincere gratitude towards my guide ***Prof. Sridhar Iyer*** for his constant support, motivation and guidance. Without his deep insight into this domain and his valuable time for this project, it would not have been possible for me to move ahead properly. He has been remarkable in his attempt to keep me motivated in this project and has always tried to improve me with proper feedback.

I would like to express my indebtedness towards ***Prof. Sahana Murthy*** for her suggestions and invaluable support throughout the project.

I would like to thank my friends ***Neelamadhav G*** and ***Vijay Kumar*** for their constant feedback and motivation regarding this project.

I would like to thank each and every one who helped me throughout my work.

Souman Mandal
(09305066)

Abstract

Problem Based Learning (PBL) is a student centric teaching-learning strategy. In PBL students solve a problem or problems in a group to achieve the learning objectives(LO). Many research have shown that PBL is a very effective instructional strategy, particularly for the educational field like Medical and Engineering, where students need to apply their knowledge to solve real life problems. But the implementation of PBL is time-consuming and sometimes student and facilitator both find it hard to begin with. Assessment, communication, judgement of student progress are also difficult part for successful outcome in a PBL course. So a well-structured platform is required to support the whole process of a PBL course. Existing Learning Management Systems (LMS) or Course Management System (CMS) like Moodle can be used to manage PBL courses, but these LMSs are very general. So to get more effective results a tool developed based on the pedagogical philosophy of PBL is needed. As these LMSs are highly used by the universities, schools, other organizations and some of the existing features of these LMSs can be reused, it is better to build the PBL tool as an add-on for an existing LMS. Thus, to support PBL we have developed a plug-in for Moodle as an activity module. For assessment purpose another module named Rubrics is created. In this report different features of existing LMSs has been explored, different steps of PBL has been described, internal structure and features of Moodle are depicted. We also proposed a system which can support each of the steps of PBL and finally we described the user documentation and developer documentation of the PBL and Rubrics module.

Contents

List of Figures

List of Tables

Acronyms and Abbreviations

- **PBL** : Problem Based Learning
- **LMS** : Learning Management System
- **CMS** : Course Management System
- **RLF** : Relevant Learning Facts
- **LO** : Learning Objective
- **API** : Application Programming Interface
- **GUI** : Graphical User Interface

Definitions

- **PBL:** Problem based learning is a student-centered instructional strategy where student solve problem working collaboratively in a group.
- **Blended Learning:** Blended learning is combination of traditional face-to-face learning and on-line learning using some technology.
- **LMS:** It is a software tool, which helps to deliver course content, manage courses.
- **RLF:** RLF are the learning facts which are necessary to know to solve the problem given in a PBL course.

Chapter 1

Introduction

Popularity and implementation of distance learning is increasing rapidly. Only The Open University,UK which offers most of its courses as distance learning course alone, has more than 168,850 registered students[6]. In US in 2006, more than 96 percent of the largest colleges offered on-line courses[8]. This two data clearly shows that distance learning is widely used now-a-days. In distance learning, people can study by their own pace, learning is not restricted by time and distance; means people don't need to stay at the university campus, they can study in the time they feel comfortable. So distance-learning has some advantages. To support distance education use of technology is must. A large number of course management system(CMS) or learning management system(LMS) softwares are there. This softwares mainly helps to manage courses, deliver course contents, conducting on-line exam etc. Now if we want to implement a PBL course in distance learning environment, a PBL module or tool which support the steps of PBL in more structured way than existing LMSs and address the implementation difficulties of PBL, then it will be helpful. Here supporting a step means, providing the infrastructure, so that user can easily do his/her task in that step. Like, problem presentation by the facilitator, is one of the steps in PBL. So, the tool can support this step by providing a HTML editor to help the teacher to publish the problem statement. Now this tool will help to execute a PBL courses offered in distance learning environment. For this the tool need to be complete; means it should have some support for each step of PBL. Now the questions are, how to develop the tool, what features will the tool have, how teacher or student will get benefited using this tool etc.

The tool to support PBL has been developed as a plug-in for the learning management system Moodle. There are two basic reasons behind this, one is the popularity of Moodle and the other is the features which Moodle already provide. There is a module named Project in Moodle[2], which was created to support projects student carry-out in a course but it doesn't go with the pedagogical philosophy of PBL. There are examples of courses which have been carried out with the help of Moodle[20], but some of them complained that carrying out a PBL course is difficult and time consuming, tracking progress is difficult etc.

The rest of the report is organized as follows. What is Problem Based Learning is described in chapter 2. A comparative study among the different LMSs and why Moodle is better is written in chapter 3. Related works about developing systems to support PBL is described in chapter 4. Proposed system is described in chapter 5 and 6. Integration plan is stated in chapter 7 and chapter 8 is about the Moodle internal structure and features, developer documentation and user documentation of the rubrics module is in chapter 9, description of developer documentation and user documentation of the PBL module in chapter 10 and finally the conclusion is in chapter 11.

Chapter 2

Problem Based Learning

Problem based learning (PBL) has been used in higher education for a long time. As a model PBL was developed for the medical education in the early 1970's. It was implemented first in McMaster University, Canada[10].

2.1 Pedagogy behind PBL

Generally here in India and in most of the world up-to 12th standard we are taught in traditional directed teaching methodology. Even in graduate level the scenario is almost the same. But cognitive science research in education shows that student constructs their knowledge by their own, they do not take it in as it is delivered directly. Some time we learn better while working in a group by peer teaching and it also helps improve the communication skills. So a student can remember and apply their knowledge better if the learning is student centric, collaborative and discovery based. For education like medical, engineering, business administration, problem solving skill is highly required as students have to apply their knowledge directly to solve real-life problems. But traditional teaching method doesn't provide that platform. To keep-up with the rapidly changing world of engineering and medical science another skill is also very much required, i.e. learning *how to learn*.

Now Problem Based Learning is a student-centric teaching methodology in which student solve one or more problem (preferably real-world) in a small group and teacher works as a facilitator. So problem based learning promotes *"active, constructive, contextual, co-operative, and goal-directed learning"* [16]. Various studies shows that implementation of PBL principle has positive effect on student learning process as well as learning outcomes. In the paper Effects of problem-based learning: a meta-analysis[9], authors reviewed 43 articles, which are about empherical study of problem based learning, and they concluded that *"there is a robust positive effect from PBL on the skill of the students"*

2.2 Steps of PBL

Though there are some variations in implementation of PBL in different courses (Like: in some implementation there are different roles given to the students with in a group as Student, Chairperson, Speaker etc[15], where in other implementation there were no such division), the steps in the process of PBL are almost the same. In the papers [10] and [13] authors recommended six steps for PBL. Schmidt and Moust proposed seven-jump approach, which are basically guidelines for students activity in PBL. Now if we count the steps, starting from group formation and facilitator allocation to evaluation by the facilitator then the following 12 steps are commonly followed steps in PBL.

1. Students are divided into groups
2. One facilitator is assigned to each group
3. Facilitator presents the problem to the groups
4. Student discuss among themselves and with the facilitator to have a better understanding about the problem
5. Students identify what relevant fact they already know and what they need to know to solve the problem
6. Student/facilitator search/share/evaluate resources which can be useful to learn about the problem domain
7. Student reads(books, documents, shared resources) /learns(can be done by doing some experiments) to gain knowledge about the problem domain
8. Students propose solutions and discuss different aspect of it with-in the group
9. Ultimately they come up with the best solution they can think
10. Each group present the solution
11. Student self-evaluate themselves, and peer evaluate others in the group
12. Teacher does the final evaluation

Here step 4 to step 8 can be repeated according to the need of the group and the problem. This 12 steps include all the steps recommended in[10, 13] and student will be able to follow the seven-jump approach. To validate this, we can compare this twelve steps with the, six steps recommended by Terry Barrett in the paper *Understanding Problem-Based Learning*[10]. This steps are,

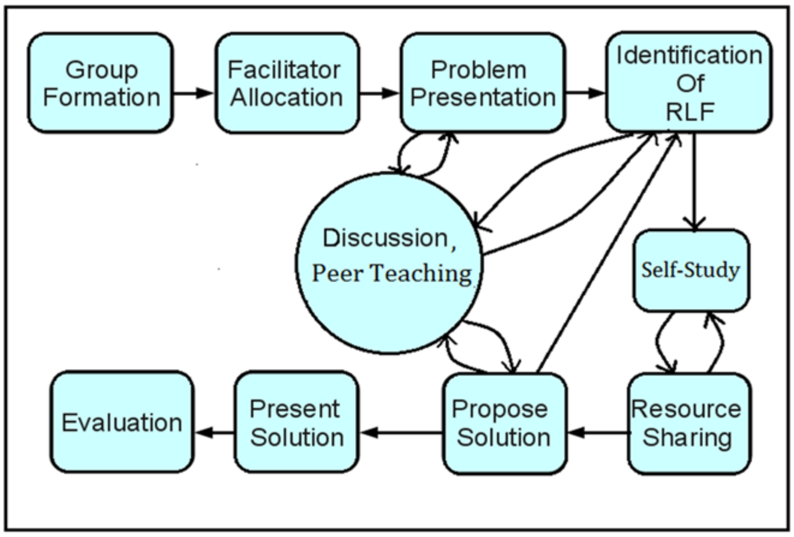


Figure 2.1: PBL process

1. First students are presented with a problem.

2. Students discusses the problem in a small group PBL tutorial. They clarify the facts of the case. They define what the problem is. They brainstorm ideas based on the prior knowledge. They identify what they need to learn to work on the problem, what they do not know (learning issues). They reason through the problem. They specify an action plan for working on the problem.

3. Students engage in independent study on their learning issues outside the tutorial. This can include: library, databases, the web, resource people and observations.

4. They come back to the PBL tutorial(s) sharing information, peer teaching and working together on the problem.

5. They present their solution to the problem.

6. They review what they have learned from working on the problem. All who participated in the process engage in self, peer and tutor review of the PBL process and reflections on each person's contribution to that process.

In this *six steps* it is assumed that first two steps of the *twelve steps* described in this paper are already executed. Step 2 of the *six* contain, step 4 and 5 of the *twelve steps*. Step 6 and 7 in *twelve steps* approach are same as step 3. In the same way other steps of this *six* are same with the rest of the *twelve steps*.

Now automatically the question comes that why we need break some of the six steps and makes it a twelve steps process. The answer is, this break up will ease the task of technical implementation of tool to support a for PBL process. Like for example if we take step 2 of *six step* process, then discussion and keeping record of different learning objective can be implemented using two different modules. So it is divided into step 4 and 5 in the *twelve steps* method.

2.3 Role of Teacher in PBL

In PBL role of teachers is different from the traditional teaching method. Teachers in PBL act like facilitators. They will stimulate the student in their constructive experience. They offer the resources, cases, courses and teach the way of gaining resources, collecting and analysing data [23]. But they never directly give the way to the solution using the knowledge they have on the domain of that particular problem. To challenge the learner's thinking they might constantly ask: *"Do you know what that means?"*, *"What are the implications of that?"*, *"Is there anything else?"*. So in this way facilitator increase the creative thinking of the student. In problem based learning the teacher act like a catalyst or enzyme [3].

$$S + TK \rightarrow SK + TK$$

$$S + K \rightarrow SK$$

$$T$$

Here S denotes Student, T denotes Teacher and K stands for Knowledge.

2.4 Advantages of PBL

There are many examples of implementation of PBL in courses. Most of them shows that PBL is better teaching methodology than traditional teaching[18, 21, 20, 15, 11]. Learning through problem solving is more effective than memory based learning.

- **Problem-solving and Research skills:** Student develop the skill *"how to learn"* and skill of critical thinking[14]. Student also develops the skill of self-directed learning.
- **Motivation:** As the problems are real world problems and PBL has more motivational appeal than traditional methods[15].
- **Social skill:** Student also adopt themselves in a team-working environment which is very much vital to work in professional organizations.
- **Effective:** Now when comes the situation where students have to apply their previous knowledge, student taught in PBL methodology, apply it better.

2.5 Disadvantages of PBL

- **High teacher-student ratio:** If the teacher-student ratio is low then large tutorial group need to be formed. In a large group it will be difficult for the facilitator to facilitate the group, track performance of individual student. So PBL require high teacher-student ratio.
- **Cost:** As implementation of PBL course require more time than traditional methods, if teacher is new to PBL then he might need training, students need access to different documents, infrastructure for discussion etc, PBL is a costly process.
- **Effect on Knowledge:** According to[9] in PBL course students *gained slightly less knowledge* than traditional teaching methodology, though they *remember more of acquired knowledge*.

2.6 Challenges in PBL

- **Different:** PBL is a different than traditional methods, so if students are never taught in PBL method they find it hard to switch[16, 18].
- **Assessment:** Traditional assessment do a little in PBL[25]. Most student feel insecure in PBL if traditional assessment is followed[24]. PBL is a process based approach than product based. So the teacher should not judge based on only the final report, rather should evaluate a student based on participation and contribution throughout the whole PBL course.
- **Free Riders:** Free riders is a term used in PBL, to indicate students who doesn't contribute in the group. Tracking these students is a challenge in PBL.
- **Time Consuming:** Teacher need to do lots of extra work in PBL, beside the subjective support to the student. Works Like group formation, tracking the progress of the group, determining strategy for self, peer evaluation and process the result of peer and self evaluation etc.
- **Communication:** In PBL student work in group, so generally lots of communication happens. So proper communication tools are required.

2.7 How technology can help?

Now technology can help for each of the challenges described in the last section. For example let's take the task of peer and self-assessment. So if we do the task manually then, this steps followed are:

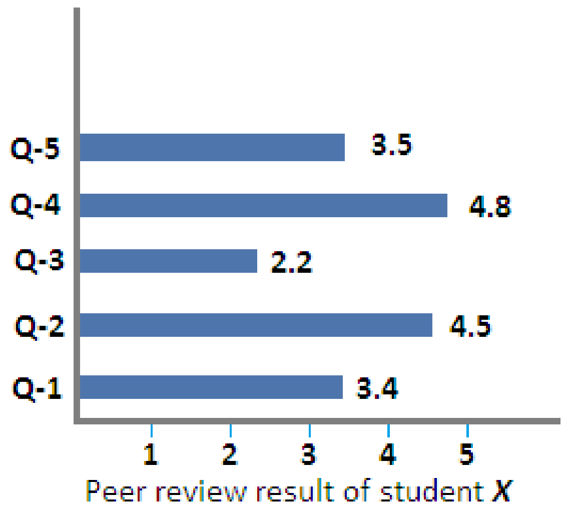


Figure 2.2: Peer review result

- Facilitator creates the forms, using some software or pen and paper.
- Then have to print it and share it with the students.
- After that students give the feedback.
- Then if the facilitator wants to find out peer feedback result for a particular student, he needs to find all the form belong to that particular student.
- Then the facilitator have to process feedback from each peer for each question.

This effort can be easily reduce by creating a web interface, where teacher can create the rubrics and students will be able to give feedback and submit the rubrics. Processing of the forms can be done automatically. So, if students have already given the feedback, then the teacher will be able to see feedback for a particular student for each question. Figure 2.2 shows a demo peer feedback result of student **X**, where the rubric contains 5 question and for each question there is rating from 1 to 5.

Details of how technology can help, specifically in the context of the proposed system is given in section 6 after the proposed system is described.

Chapter 3

Moodle and other LMSs

There are many open source and proprietary LMSs available. Moodle, aTutor, Claroline, Dokeos, HotChalk, CCNet, Blackboard Learning System, Angel, Sakai, JoomlaLMS are some of the examples.

3.1 Different features of LMSs

Now, without using any LMS, universities around the globe has run courses for years, so now-a-days why there is so much of use of Learning management system. The answer is the features an LMS provide. Table 3.1 shows the different features, generally one LMS have.

Feature	**Description**
Communication	For communication different tools are provided in a LMS. Like, for announcement news forum, synchronous communication forum, wiki, blog and for asynchronous communication blog etc.
Publishing/sharing documents	Most of the LMS provide easy interface to publish and share document in different format. User can use the inbuild HTML editor to create content.
Calendar	This feature helps user to keep track of what is going on and upcoming events in the course. So basically it helps to manage the course
On-line Exam	LMSs can be used for online exams. Generally LMSs support different question pattern, like: multiple, numerical, short essay etc.

Table 3.1: **Different features of LMSs**

So using this features teacher can manage their course better and can save time.

3.2 Moodle

Moodle is an open-source e-learning tool. The project to build Moodle was, started on ’90 s and Moodle 1.0 was released on August 20, 2002. It was created by Martin Dougiamas, at Curtin University in Perth, Australia. The term Moodle is an abbreviation of *Modular Object-Oriented Learning Environment*. Among the open-source LMSs, Moodle is being used by a large number of users and it is having more features, compared to some other LMSs[7]. Moodle have almost 50,000 registered sites and more than 36 million of users of [4]. Moodle has support for more than 90 languages and there are almost 270 people[4] worldwide working on the development and translation of Moodle. The Moodle community is also very active.

It is possible to add new modules in Moodle. There are almost 650 plug-ins already available[4], and it is possible to develop plug-ins for Moodle for different purpose. Like a developer can create a new activity module, block, question type etc.

3.3 Comparative study

In the paper *Why Moodle*[7] authors have done a comparative study among Moodle 1.8, Blackboard Learning System (V.7), ANGEL Learning Management Suite (7.1), eCollege, Claroline 1.6, Dokeos 2.1.1 and Sakai 2.3.1. Authors have compared on both technical and usability features of these tools. Among 40 different usability features authors find that Moodle 1.8 and Sakai 2.3.1 are the most complete ones, both of them having support for 38 features out of 40. Claroline is the weakest one having only 32 of 40 features. Though Sakai and Moodle having support for same number of features, Moodle is better when documentation, security, support etc are taken into consideration.

Chapter 4

Related Works

To support PBL different tools have been developed. Some of them are developed to support a particular step of PBL(like web-based evaluation[12]), others support multiple aspect of PBL.

- In Moodle there is a plug-in *Project Based learning Module*. But it has only 5 steps

  - Brainstorm
  - Signup
  - Submit
  - Schedule
  - Assessment

  This plug-in was mainly developed to manage general student projects, where students need to submit some file(.HTML) at the end of the project. So it was not developed based on the pedagogy of PBL.

- In the paper *Supporting Collaborative, Problem-Based Learning Through Information System Technology*[17] authors describe and analyze a system called CoMMIT(Collaborative Multimedia Instructional Tool-kit). The authors have implemented this WWW based software to in a PBL course to evaluate its performance. There are three module of this software, *Authoring*, *Instructor-support* and *student module*. *Authoring module* is basically a graphical language, which enable the user to graphically connect different documents as a precedence graph. Where to study about a document student should have knowledge about the precedence nodes of the document. *Instructor-support module* produce some graphs or other kind of data to help instructor to monitor and evaluate groups. *Student module* allows the student to access the documents in a controlled manner. So that they always reads the prerequisites before any document. It also allows the student to take general notes.

  This system helps the facilitator and student in some of the steps in PBL. So this is not a complete system. This is also a standalone system.

- In another paper *Designing Web-based Interactive Learning Environments for Problem-based Learning*[22] authors describe a system named INDIE. It is a web-based learning environment. It provides an *authoring tool* and a form based learning authoring tool and a *run-time engine*. This web-based learning environment provides, a *welcome interface* showing problem definition, *reference interface* where documents relevant to problem domain can be found, an *experiment interface* where students can simulate their experiments, a *feedback interface* where teacher can provide feedback and a *report interface* where student can write the report for the problem or experiment.

  This is also a standalone system. It does not support collaborative learning.

- In the paper *Tools and strategies for improving PBL laboratory courses with a high student-to-Faculty ratio*[19] authors developed multiple tools to support administration and teaching-learning. Automated web-based student enrolment, Laboratory slot management, Student survey management are different tool to ease the task of administration. For teaching-learning authors developed tools for student progress management, e-mail based communication, plagiarism and cheating detection etc.

  Now the tools which authors have developed are all independent; means they are not been incorporated in a single software package. So if we want to use them we need to install each of them individually. Support for collaborative work is also less.

Chapter 5

Proposed System

The proposed system will help the instructors to run their PBL courses in both distance learning and blended learning environment. It will be integrated with Moodle as a plug-in. So it need to be developed by following the rules, methods and technology which are necessary to build a plug-in for Moodle. Moodle is mostly used to manage different type of courses. Here a teacher can add different activities as part of their course, like: quiz, wiki, assignments, survey. Students need to carry-out these activities. PBL module will also be one type of activity, which will come under the ***Add an activity...*** button. Figure 5.1 and 5.2 shows how in Moodle PBL module will be integrated.

5.1 User characteristic

Both teacher and students must be familiarized with basic handling of computer, internet browsing to use this new PBL module.

5.2 Operating Environment

This module will be developed using PHP, HTML, MySQL, JavaScript and CSS. So this module will run on different operating environments, which support these technologies and where Moodle runs.

5.3 Users of the system

In this system there are three different types of users, Teacher, Facilitator and Student. Their roles are defined in the Table 5.3

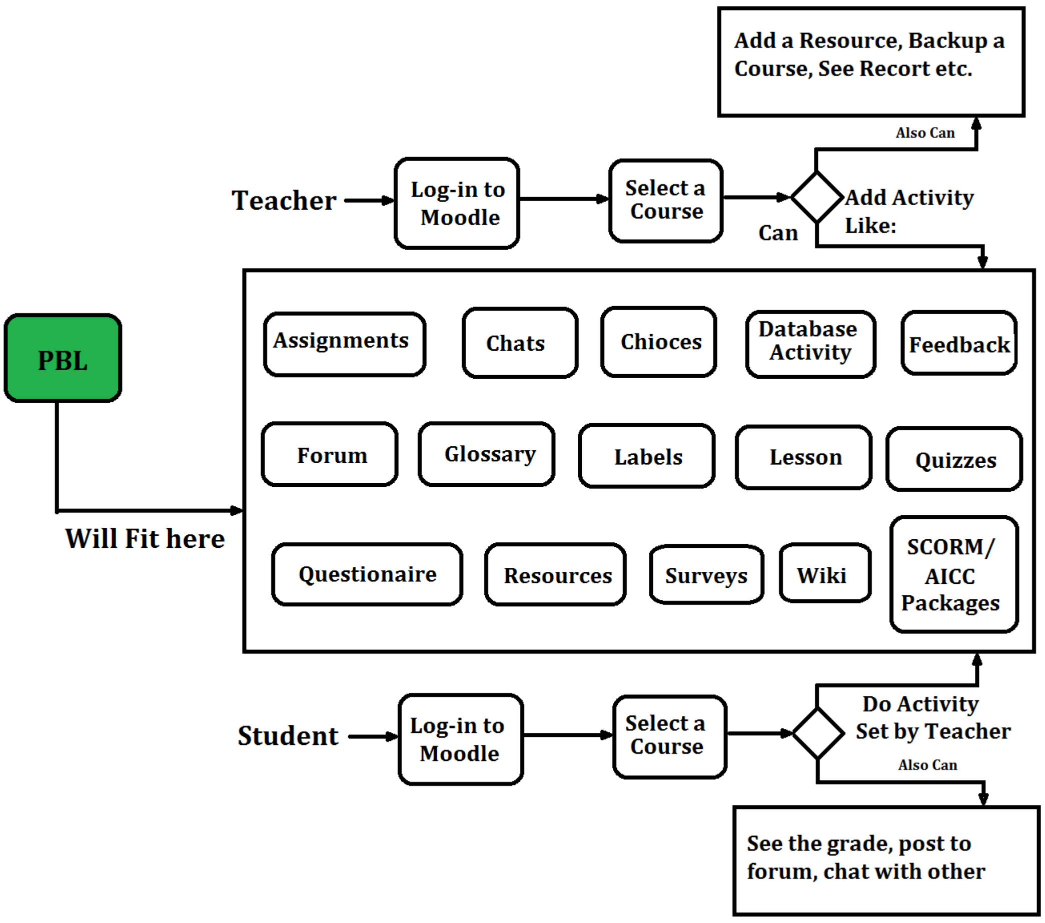


Figure 5.1: PBL activity module in Moodle

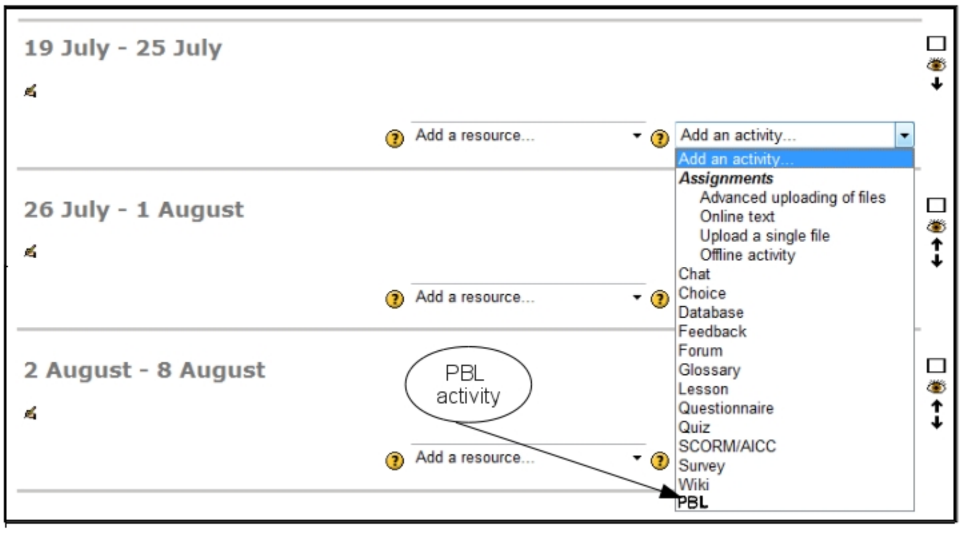


Figure 5.2: Add PBL activity module in Moodle

User	Role
Teacher	Teacher is the person who is the instructor for the course, where PBL activity can be added. Teacher will be able to see the activities of all the groups. Teacher can do all the activity, which a facilitator can do. In addition to that teacher will be to do two task. First one is dividing students in groups and the other one is allocation of facilitator for the groups.
Facilitator	Facilitator is the person who facilitates one or more than one group, in a PBL course. A teacher can also be a facilitator. Facilitator will be able to assign task and see all the activities of the groups he/she is facilitating.
Student	Student is one, who is attending the course where PBL is an activity, and has been assigned as a member of some group. Student will be able to communicate and see the activities of others in his/her group and do tasks to complete the PBL course.

Table 5.1: **Different Users of the system**

Chapter 6

Functional Requirements

In this chapter we described the different functionalities of a tool, which can support all the steps of Problem Based Learning. This PBL module will be an activity within a Moodle course. This activity will have several sub-activities.

1. Group Formation
2. Facilitator Allocation
3. Problem Presentation
4. Discussion
5. Identification of Relevant Learning Facts(RLF)
6. Resource sharing
7. Propose Solution
8. Submit Solution
9. Evaluation

Besides this sub-activities there will be some sub-modules also.

- Summary
- Recent Activity

The System is described in the following format. Each subsection heading is one feature. *User*, is the person who can use the feature. In *Description* the feature is described. *Dependency*, is the prerequisite work need to be done, before a user can use this feature, and *Rating* define whether the feature is an essential one or optional.

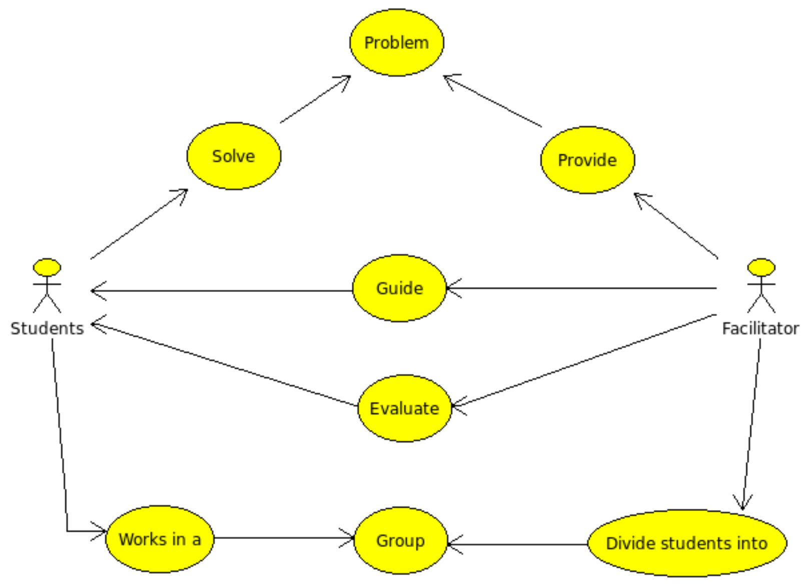


Figure 6.1: Overview of PBL process

6.1 Group Formation

6.1.1 Automatic Group Formation

User: Teacher
Description: Teacher will give either the number of the groups or the number of members with-in a group and it will create the groups automatically. This automatic group formation can be done randomly, or taking into account some constrain like: alphabetical order of name, or in a online course, common-time which is most suitable for the students to work together.
Dependency: All the students and faculty has registered in the Moodle course in which PBL is an activity.
Rating: Essential

6.1.2 Manual Group Formation

User: Teacher
Description: Teacher will be able to form groups of any size $\leq$ total number of student in the course, with any member $\leq$ total number of student in the course.
Dependency: All the students and faculty has registered in the Moodle course in which PBL is a activity.
Rating: Essential

6.1.3 View Group

User: Teacher
Description: Should be able to view who are the members of which group. Student will be able to see the information of other group only if Teacher has enabled the option to see the details of other group's info.
Dependency: After group allocation is done.
Rating: Essential

6.1.4 Group Permission

User: Teacher
Description: Should be able to set permission whether a student can see the activity of other group or not.
Dependency: After group allocation is done.
Rating: Optional

6.1.5 View Members

User: Student
Description: Should be able to view who are the members of his/her group.
Dependency: After group allocation is done.
Rating: Essential

6.1.6 Change Group

User: Teacher
Description: Using this the teacher should be able to change the group formation, remove a student from the group, add a new student or swap students.
Dependency: After group allocation is done.
Rating: Optional

6.2 Facilitator Allocation

6.2.1 Assign Facilitator

User: Teacher
Description: For each group at-least one facilitator should be allocated. One can be facilitator for more than one group.

Dependency: This step need to be executed after the group allocation is over.
Rating: Essential

6.2.2 View Facilitator

User: Student
Description: Should be able to view the facilitator assigned to his group.
Dependency: After facilitator allocation is done.
Rating: Essential

6.2.3 Groups I'm Facilitating

User: Facilitator
Description: Should be able to view which group or groups he is facilitating.
Dependency: After facilitator allocation is done.
Rating: Essential

6.2.4 Change Facilitator

User: Facilitator
Description: Should be able to change facilitator for a group.
Dependency: After facilitator allocation is done.
Rating: Optional

6.3 Problem Presentation

6.3.1 Problem Definition

User: Facilitator
Description: Should be able to present a problem to each group. Facilitator can present the same problem to all the groups or can present different problems to different groups.

- Facilitator should be able to present the problem in different ways. Facilitator should be able to submit a text, with which facilitator can add different multimedia file video, audio, PPT , Document files, animation etc.
  Rating: Essential
- Video conferencing with the group to present the problem.
  Rating: Optional

Dependency: After facilitator allocation step done.

6.3.2 View Problem

User: Facilitator
Description: Should be able to view the problem they have posted.
Dependency: After the problem presentation done.
Rating: Essential

6.3.3 View Problem

User: Student
Description: Should be able to view the problem assigned to them.
Dependency: After the problem presentation is done.
Rating: Essential

6.3.4 Edit Problem

User: Facilitator
Description: Should be able to edit the problem definition they have presented.
Dependency: After they have presented the problem.
Rating: Essential

6.3.5 Comment on the problem statement

User: Student/Facilitator
Description: Should be able to post question/doubt related to the problem statement, and the answer to this doubts.
Dependency: Problem presentation is done.
Rating: Essential

6.4 Discussion

6.4.1 Add Discussion

User: Facilitator
Description: Facilitator will be able to add discussion when ever needed. Discussions will be mainly of two types: Asynchronous and Synchronous. For asynchronous discussion there will options like: Forum, Wiki etc. For Synchronous discussion there will be option for, text-chat, audio-chat, video-chat, whiteboard. Options for forum, wiki, text-chat is essential. Other options are not essential.

Dependency: After facilitator allocation is done.
Rating: Essential

6.4.2 Chat

User: Student/Facilitator
Description: Students can chat with the on-line group members and facilitator. Facilitator can chat with the on-line members of the groups he/she is facilitating.
Dependency: After facilitator allocation is done.
Rating: Essential

6.4.3 Rate

User: Facilitator/Student
Description: Facilitator will be able to rate the questions or reply given by students in a forum. Students will also can rate replies given by other group member. This rating will be in a scale of 1-5.
Dependency: After a question is asked or replies given in a forum.
Rating: Essential

6.5 Identification of RLF

6.5.1 Define RLFs

User: Student
Description: Student should be able to document different RLFs which are needed to solve the problem. For this, one interface should be provided such that they should be able to ***list*** what they already know and what need to know to solve the problem.
Dependency: Problem presentation is done.
Rating: Essential

6.5.2 Comment on RLF

User: Facilitator/student
Description: Should be able to post comments on the student RLFs.
Dependency: After Identification of RLF is done.
Rating: Optional

6.5.3 Edit RLF

User: Student
Description: Should be able to change the RLFs.
Dependency: After identification of RLF is done.
Rating: Essential

6.5.4 View RLF

User: Facilitator/Student
Description: Facilitator should be able to view the RLFs of the members of the group of which he is the facilitator. Student should be able to view his and all other's (who are in the same group) RLF.
Dependency: After identification of RLFs.
Rating: Essential

6.6 Resource Sharing

6.6.1 Share Resource

User: Facilitator/Student
Description: Facilitator should be able to share resources with a particular group or all the groups he is facilitating. Student should be able to share resources with the group and the facilitator. These resources can be any multimedia file, like: picture, PDF file, video, audio, .doc, text, etc or it can be a link to some web document, or can be a name of a book also. A shared document might help to know about some of the RLF. So, for the document there should be a facility to mark which RLF it will help to learn. Whenever facilitator shares a document a pop-up window will come asking him to mark RLFs. This list of RLF will be the union of the list of identified RLF by each individual in the group. In the same pop-up window there should be an option in which owner would be able to give description of the resource.
Dependency: After RLF identification is done.
Rating: Essential

6.6.2 My Folder

User: Facilitator/Student
Description: Should be able to store and manage all the document user has shared or want to share or for his own study/reference.
Dependency: This should be the same as the Moodle Files module.
Rating: Optional

6.6.3 View Shared Resources

User: Facilitator/Student
Description: Should be able to view all the resources user has shared and with whom.
Dependency: After RLF identification is done.
Rating: Optional

6.6.4 Search Resources

User: Facilitator/Student
Description: This will produce result using the document descriptions, RLFs marked for the documents, or texts (if the document has text in it like: .txt, PDF, html etc) of the shared documents.
Dependency: After Resource Sharing is done.
Rating: Optional

6.6.5 Rate Resource

User: Facilitator/Student
Description: Facilitator will be able to rate the resources shared by students and him/her. Students will be able to rate the resources shared by other group members and facilitator.
Dependency: After some resource is shared.
Rating: Essential

6.7 Propose Solution

6.7.1 Solution Proposal

User: Student
Description: Student should be able to propose a solution to the problem and share it with the group members and facilitator will also be able to view the solution.

- Student should be able to present the problem in different ways. Student should be able to submit a text, or a weblink, with which he can add different multimedia file video, audio, PPT, Document files, animation etc.
  Rating: Essential

- Video conferencing with the group to present the problem.
  Rating: Optional

- Audio conferencing with the group to present the problem.
  Rating: Optional

Dependency: After the problem presentation is done.

6.7.2 Another Solution Proposal

User: Student
Description: Student should be able to propose more than one solution to the problem.
Dependency: After one solution is proposed.
Rating: Essential

6.7.3 View Proposals

User: Student/Facilitator
Description: Student should be able to view the proposed solution by the other members of the group. Facilitator should be able to view the proposed solution by the members of the groups which he is facilitating.
Dependency: After the problem definition is done.
Rating: Essential

6.7.4 Comment on Proposal

User: Student/Facilitator
Description: User should be able to post comment about the proposed solutions by others/own.
Dependency: After solution proposal is done.
Rating: Optional

6.8 Submit Solution

6.8.1 Submission Guideline

User: Facilitator
Description: Facilitator should provide a text guide line for submission of the solution. Facilitator can specify the format of the submission, like final report format, how many files etc.
Dependency: After problem presentation is done.
Rating: Optional

6.8.2 View Submission Guideline

User: Student
Description: Student will be able to view the submission guideline using this feature.

Dependency: After submission guideline is specified.
Rating: Optional

6.8.3 Edit Submission Guideline

User: Facilitator
Description: Facilitator will be able to edit the submission guideline.
Dependency: After submission guideline is specified.
Rating: Optional

6.8.4 Submission Deadline

User: Facilitator
Description: Facilitator will specify the submission deadline.
Dependency: After submission guideline is specified.
Rating: Optional

6.8.5 Submit Solution

User: Student
Description: Student will submit the solution in the specified format, as mentioned by the facilitator. There will be one final submission for the group. Final submission may contain multiple files.
Dependency: If submission guideline is there then, after submission guideline is specified or after the problem presentation
Rating: Essential

6.8.6 View/Edit Submission

User: Student
Description: Student can edit their submission until the deadline passes.
Dependency: After submission guideline is specified.
Rating: Essential

6.8.7 View Submission

User: Facilitator
Description: Facilitator should be able to see and download all the submitted document by the group/groups he/she is facilitating.
Dependency: After students submit their solution.
Rating: Essential

6.9 Evaluation

6.9.1 Create self-evaluation form

User: Facilitator
Description: Facilitator will be able to create self evaluation form using this feature. This form will be a rubric.
Dependency: Any time after problem presentation
Rating: Essential

6.9.2 Create peer evaluation form

User: Facilitator
Description: Facilitator will be able to create peer evaluation form using this feature. This form will be a rubric.
Dependency: Any-time after problem presentation
Rating: Essential

6.9.3 Create Rubric

User: Facilitator
Description: It will ask for the title of the rubric. Then number of columns and rows of the rubric. Then a table will be created. Facilitator will be able to edit the title of the column. Then can define criteria according to each row and column. This feature will be used after the facilitator selects to create the self/peer evaluation form.
Dependency: After the problem is presented.
Rating: Essential

6.9.4 Self Evaluate

User: Student
Description: Student will be able to access the self-evaluation form created by the facilitator.
Dependency: Creation of self-evaluation form is done.
Rating: Essential

6.9.5 Submit the self-evaluation form

User: Student
Description: Student will be able to submit the self-evaluation form. This can be done for only one time.

Dependency: Creation of self-evaluation form is done.
Rating: Essential

6.9.6 Peer Evaluate

User: Student
Description: Student will be able to access the self-evaluation form created by the facilitator
Dependency: Creation of peer evaluation form is done.
Rating: Essential

6.9.7 Choose Peer

User: Student
Description: Student will choose a peer whom he wants to evaluate.
Dependency: Creation of peer evaluation form is done.
Rating: Essential

6.9.8 Submit the peer evaluation form

User: Student
Description: Student will be able to submit the peer evaluation form for a particular peer. For one peer this can be done for one time only.
Dependency: Choose peer is done.
Rating: Essential

6.10 Recent Activity

6.10.1 Recent Activities

User: Facilitator
Description: This link will show the summary of the recent activities done by the students in the group the facilitator facilitating.
Dependency: After Facilitator allocation is done.
Rating: Essential

6.10.2 Recent Activities

User: Student
Description: In the PBL homepage of the student there will be summary about the updates of the recent activities of the group will be displayed. Some comparative data will also be shown

in the homepage, like:
Avg. resources shared by the individual group member are : X
Resource shared by you are : Y
Avg. rating of the discussion of the individual group member is : M
Avg. rating of your discussion is : N
Dependency: After grouping is done.
Rating: Optional

6.10.3 Show full history

User: Facilitator
Description: This will enable the user to view the whole history of the activities done by different person in the course.
Dependency: After facilitator allocation is done.
Rating: Optional

6.10.4 Show history of a particular member

User: Facilitator
Description: Facilitator will be able to choose one particular group member and then see the activities done by that member in past.
Dependency: After facilitator allocation is done.
Rating: Essential

6.11 Report

6.11.1 Result of self-evaluation

User: Facilitator
Description: Facilitator will be able to see the result of the self evaluation.
Dependency: Self-evaluation is done by the student.
Rating: Essential

6.11.2 Result of peer-evaluation

User: Facilitator
Description: Facilitator will be able to see the result of peer evaluation. Facilitator will be able to see the average peer rating of the group, of a particular student, individual rating of all the other students for a particular student.

Dependency: Peer-evaluation is done by the students.
Rating: Essential

6.11.3 Report of a individual

User: Facilitator
Description: Facilitator will choose a particular student in the group to see his report. In the report answer of the following will be there in the report. Let the student is S
Avg. number of documents shared by the group are : X
Number of documents shared by S are : Y
Avg rating of the documents shared by the group : M
Avg. rating of the documents shared by user S : N
Dependency: After facilitator allocation is done.
Rating: Essential

6.12 How new module will help better?

- There are some basic advantages of using computer added system for PBL. If we use computer support, then we can use the different multimedia functionality computer can support. In most of the cases the problem in PBL is a complex one, so while describing the problem facilitator can use animation, video, audio if required. Sometimes students also get motivated if multimedia is used.

- Developing the tool as a plug-in for Moodle is also have some advantages. In the institutions Moodle is generally used in the following manner, there is one central server for Moodle and one administrator who manages Moodle. Now if the administrator installs a new feature then all the teachers can use it. So basically teacher do not need to worry about the management, and once a plug-in is installed every instructor can use it in any course. Against this, if there is one standalone system, then either we need to use some other central server for that system, or install it in a particular system for a particular course. This increases the implementation difficulty.

- If a PBL course is offered in distance learning then use of technology tools or LMS is must to manage and deliver the course.

Now the proposed system has some particular features which help the instructor and student in a PBL course.

- **Collaboration:** In the proposed system there will be support for collaboration. Like: student can share documents with each other, can communicate in different way, list the learning objectives etc.

- **Evaluation:** As there will be support for on-line survey it will save time, compared to traditional off-line survey. It will also help the instructor to evaluate the contribution and participation of each individual for the whole PBL course by producing report.

- **Tracking:** Using this tool teacher can view recent activities and activity report of the groups he/she is facilitating as described in the previous sections. Viewing this updates, teacher will be able to decide, whether the group is working or not. This feature will also help to find out the free-riders.

- **Communication:** Besides the tools which Moodle provide for communication, white-board and audio-video conferencing are some desired feature of this tool. So these features will help in more effective communication, specially for distance learning.

  Along with these advantages, this tool will be complete, means it will help in each step of PBL. In [16] authors discussed that poor understanding of the principle of PBL and *"misguided attempts to make the approach more efficient"* can have negative effect. This tool might be helpful in this case. Because it will show an interface for both teacher and student, so that they can decide what to do in the current step and what is the next step.

Chapter 7

Features of Moodle to be used

In this chapter, for each of the functional requirement of the system what features of Moodle can be used and what new features need to be implemented or how the existing features need to be updated is described. Under each functional requirement there are two bullets. First one describe the features of Moodle can be use to support the requirement and second one describes the new features to be added or extension of already existing feature, which are required. Some time features described in the second bullet is a optional feature not a essential one.

Group Formation

- Moodle has support to create group. Teacher can create different group automatically or manually. In automatic group formation, teacher need to choose either number of groups or number of students per group and based on what criteria the groups will be formed. There are four criteria, *Randomly , Alphabetically by first name, last name, Alphabetically by last name, first name, Alphabetically by ID number*. In manual grouping, teacher need give the group name, then select the members of that group. Figure 7.1, shows the interface to manage groups in Moodle.
- The new feature could be the fifth criteria in automatic grouping. That is taking timing preference of the students into account.

Facilitator Allocation

- In Moodle we can define different roles based on the permission given to that particular role. So based on our requirement permission can be given to the Facilitator role. Figure 7.2 shows Moodle interface to create new role.
- The permission for the role of facilitator need to be define and association of a facilitator and groups he is facilitating need to be done.

Problem Definition

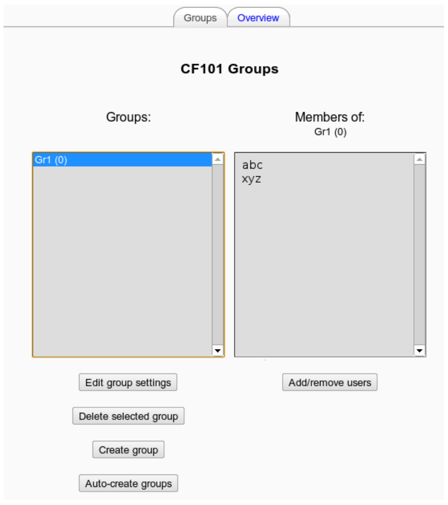
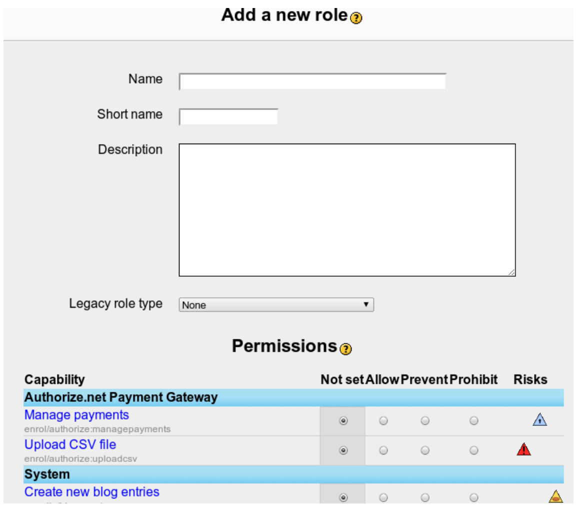


Figure 7.1: Grouping in Moodle



Capability	Not set	Allow	Prevent	Prohibit	Risks
Authorize.net Payment Gateway					
Manage payments enrol/authorize:managepayments	●	○	○	○	⚠
Upload CSV file enrol/authorize:uploadcsv	●	○	○	○	⚠
System					
Create new blog entries	●	○	○	○	⚠

Figure 7.2: Roles in Moodle

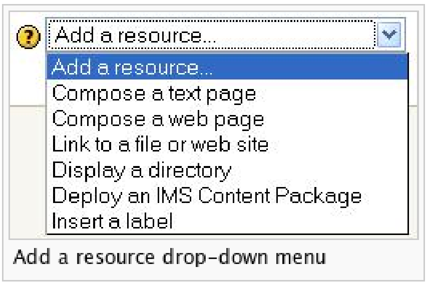


Figure 7.3: Resource sharing in Moodle

- There is a HTML, editor in Moodle, wiki or Forum. Facilitator can use this feature to present/publish the problem definition.
- A video-conferencing feature can be added, so it can be used while presenting the problem.

Communication

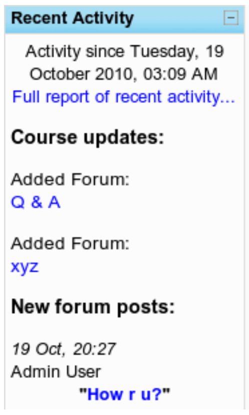
- For communication Moodle have different features. For synchronous communication, text-chat feature is there. For asynchronous communication, forum, messaging can be used.
- Addition to the features white-board, audio-video chat feature can be added for better communication.

Identification of RLF

- Currently there is no feature to record and further process the RLF in Moodle.
- New database fields and interface to list or records the RLF need to be created.

Resource Sharing

- In Moodle teacher can share different type of resources. Resource can be a file or a folder. There are options *Compose a text page, Compose a web page, Link to a file or web site, Display a directory and Add an IMS Content Package*. These features will be used to share resource in the PBL tool. Figure 7.3 shows the options for type of resources teacher can share in Moodle.
- There is no support for file or folder sharing for student in the standard Moodle software. But plug-ins[5] available which enables file-sharing facility for students. So for PBL module, either we need to use some existing plug-in or reuse the code of existing plug in, so that student can also share resources. There is also a need for associating the RLF with the shared files.


Recent Activity

Activity since Tuesday, 19 October 2010, 03:09 AM

Full report of recent activity...

Course updates:

Added Forum:
Q & A

Added Forum:
xyz

New forum posts:

19 Oct, 20:27
Admin User
"How r u?"

Figure 7.4: Recent Activity in Moodle

Propose Solution

- In Moodle support of *Forum* is there. So, student can propose their solutions in a forum as text (Different files or link can also be attached), so that other will be able to post comment on this, or students can upload files in if teacher creates an assignment activity.
- Some better way to publishing and managing the proposed solution is needed.

Submit Solution

- In Moodle teacher can add activity, where student need to upload files. So if the final solution is files to be submitted. This feature can be used.
- There need to be a feature, using which a group can present their solution. For this support for audio or video conferencing is needed.

Evaluation

- Moodle provides a feature to grade a submitted assignment. This can be used to give grade final solution.
- Standard Moodle does not provide the feature to create or process rubrics. But rubric is a necessary feature for self and peer assessment. So this feature needs to be implemented.

Recent Activity

- Moodle provide the feature, so that the user can see the recent activities happened in the course.
- For PBL we need to filter these activities which are specific to the group and relevant to PBL process.

Report

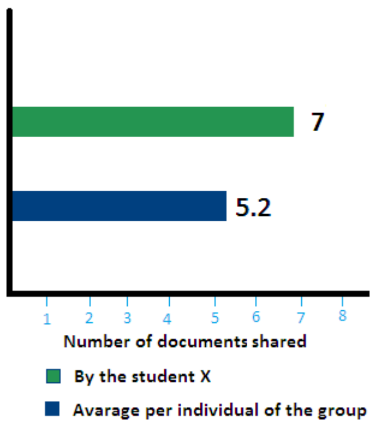


Figure 7.5: Shared resources by student **X**

- Moodle provide the feature to see the report of the all the activity in the course.
- To judge the participation or contribution of a student using the report provided by Moodle is difficult. This report must be filtered, and comparative report, which will basically show the comparison of the participation and contribution of a particular student, with others in the group. If the report is graphical, then it might help teacher to understand better. Figure 7.5, is a demo diagram. It is showing the comparative contribution for resource sharing of student **X**.

Chapter 8

Moodle Internals

In this chapter we describe the internal directory structure, database structure, access control implementation of Moodle. It is important for a developer to know about all these before start developing any new module or block. We have used some terms while describing the Moodle internals, these terms need to be explained, before we proceed. ***$HOME*** is the parent directory, where Moodle is installed. ***Course Homepage*** is the first page we get when we click on the link to a course inside Moodle. ***Module Homepage*** is the first page we get when we click on the link of a module in the course homepage. Now all the description of Moodle here is for the Moodle version 1.9.x, some of the features may be implemented in some other way in the different versions of Moodle.

8.1 Basic Structure

Moodle is an open-source Learning Management System(LMS). In Moodle we can create different format of courses, like:

- Weekly
- Topics
- Social
- LAMS course
- SCORM

Inside a course we can have different features. Most of these features are provided by some *Activity module*, like **Assignment**, **Forum**, **Quiz** or some *Block module*, like **Recent Activity**, **Latest News**, **People** etc. We can extend the features in Moodle in many ways. The two most useful ways to extend the functionalities are adding new Activity module or Block module. If we add an activity module then, it will come under the *Add an activity* 8.4 drop-down box in the course homepage. A newly added block will come in the Blocks, *Add* drop-down box in the course homepage.

The library functions are defined in the files inside the *$HOME/lib* directory. Some of the library functions are important and used frequently. This functions are defined in the file *weblib.php*, *moodlelib.php*, *dmllib.php* and *accesslib.php*.

8.1.1 Library Functions

- All the function to update, insert, delete database entries in Moodle are defined in the *$HOME/lib/dmllib.php* file. For example the function

  - ***delete_records($table, $field1, $value1, $field2, $value2, $field3, $value3)***

  is used to delete records from a database table. Here *$table* is the name of the table from where we want to delete record/records. *$field1* is the first field to check and *$value1* the value field1 must have. Same applies for *$field2*, *$value2* and *$field3*, *$value3*. In this particular function, only the *$table* parameter is mandatory, others are optional.

- General purpose Moodle functions are defined in the *$HOME/lib/moodlelib.php*. For example the function

  - ***required_param($parname, $type)***

  returns a particular value for the parameter named *$parname*, taken from POST or GET method. If the parameter does not exist then an error is thrown. Here *$parname* is the name of the parameter and *$type* is the type of the parameter.

- Library of functions for web output are defined in the *$HOME/lib/weblib.php* file. For example the function

  - ***print_header ($title, $heading, $navigation, $focus, $meta, $cache, $button, $menu, $usexml, $bodytags, $return)***

  prints a standard header in the page, where it is being called.

- To control access functions are defined in the *$HOME/lib/accesslib.php* file. For example the function

  - ***has_capability($capability, $context, $userid, $doanything)***

  checks whether a user has the capability with regards to the context. Here *$capability* is the capability name, *$context* is the context of the access, *$userid* is the id of the user and if *$doanything* is false then it gets ignored, otherwise it will check for the permission.

8.1.2 Blocks and Modules

In Moodle all the blocks are defined inside the *$HOME/block* directory. All the activity modules are defined inside *$HOME/mod* directory. Files which are required to show a course and add or delete a particular module in a course are defined inside the *$HOME/course* directory. If we see the course homepage, then *$HOME/course/view.php* is shown. If we try to add a particular module or resource then we will be redirected to *$HOME/course/modedit.php*. Now when ever in a course(course type is *week*) we try to add a particular module, 5 variables are passed using the GET method to the *$HOME/course/modedit.php*. These are *add*, *type*, *course*, *section*, *return*. Here *add*, is the name of the module we wanted to add, *type* is the type of the module, *course* is the course id, *section* is the section number where we wanted to add the module and depending upon the *return* value, Moodle decides where to redirect after adding the module. To update, move, delete a module Moodle use *$HOME/course/mod.php* file.

8.1.3 Moodle QuickForm

Moodle use the HTML form extension QuickForm for easy definition, process and validation of form. We can explain, how Moodle use the QuickForm using one example. The code given below is the code for the form which we used to take the input for RLFs in the PBL module. Lets call this *RLF form*.

```
class rlf_form extends moodleform {
  function definition() {
    global $COURSE;
    $mform = & $this->_form;
    $mform->addElement('header', 'general', get_string('general', '
        form'));
    $mform->addElement('text', 'name', get_string('rlfname', 'pbl')
        , array('size'=>'64'));
    $mform->setType('name', PARAM_TEXT);
    $mform->addRule('name', 'This is a required element', 'required
        ', null, 'client');
    $mform->addElement('text', 'link', get_string('rlflink', 'pbl')
        , array('size'=>'64'));
    $mform->addElement('hidden','user_id');

     $this->add_action_buttons($cancel = false);
  }
}
```

In this RLF form, we define a class *rlf_form* which is an extension of the *moodleform* class. Now this *moodleform* is the base class for the QuickForm. Whenever we write a form using the QuickForm extension, we need to use this *moodleform* as the base class. There are different type of functions define in the newly created *rlf_form* class. To add any element in the form we can call the **addElement()** function. For example here we have used the

- ***addElement('text', 'name', get_string('rlfname', 'pbl'), array('size'⇒'64'))***

function, *'text'* is the type of the element, *'name'* is the name of the element. Using the *get_string()* function we fetch the text which will be shown before this text element from the local language file. For the PBL module the local language file is located in the *$HOME/mod-/pbl/lang/en_utf8* directory and the name of the file is same as the module, *pbl.php*. The last argument, *array()* is to restrict the size of the textbox. In this *rlf_form* we have added three elements.

Now for each element we add, we can put different constrain using the **addRule()** function. For example here we have used,

- ***addRule('name', 'This is a required element', 'required', null, 'client')***

Here *'name'* is the name for the element we want to add rule. The second element is the message user will get, if he do not fulfill the constrains. Third element is the type of constrain, which is *required* here. The forth element is for extra rule and the fifth element decides where the form validation will take place. So for this form RLF name is a required element, user can not left it blank.

To set the default value for a particular element we need to use the **setDefault()** function. The prototype of this function is given below.

- ***setDefault(element_name, value)***

Here *element_name* is the name of the element we want to set a default value, and *value* is the value we want to set as default.

To add the standard action button for the form we need to use the ***add_action_buttons()*** function. To display this particular form we need to create an object of the class *rlf_form*, and call the display function of the class. So, let the object name is *rlf_Form*, then we will call ***rlf_Form→display()*** function. Instead of writing *addRule()* function, the alternate is to write a function inside the class named *validation()*, to validate the data.

8.1.4 Configuration

Inside the *$HOME* folder we can find the Moodle configuration file, *config.php*. In this configuration file all the configuration settings regarding the database and different configuration variables are stored. Among these variables, three variables are used frequently while developing the PBL and the rubrics module. These are,

- ***$CFG→wwwroot***: Root web address for Moodle.
- ***$CFG→dirroot***: Root directory address for Moodle.
- ***$CFG→dataroot***: Root data directory address for Moodle.

8.1.5 Moodle XMLDB editor

The XMLDB editor helps to set up the database tables of Moodle, by creating the .xml files. This editor basically serves two purposes.

1. Moodle supports different databases, like: MySQL, PosgreSQL etc. So depending upon the database, code to update the database will be different. Now to update different databases developer need to write different code. This redundancy of writing different code can be avoided using the XMLDB editor.

2. Updating database using the XMLDB editor is easy and this editor also automatically updates the install.xml file.

8.1.6 Using the XMLDB editor

If we want to add a new filed in some table of Moodle database, we can do it using the XMLDB editor provided in Moodle. All we have to do, is to follow the steps defined below,

1. Log into Moodle site as Administrator

2. Go to the left-bottom link *Miscellaneous* as shown in figure 8.1(a).

3. Click on the link *XMLDB editor* as shown in figure 8.1(b).

4. *Load* the database and click on *edit* as shown in figure 8.1(c).

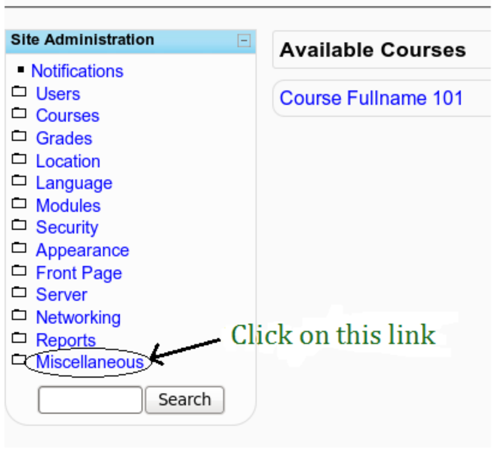
5. Now we have to select the particular table we want to update or if required we can add new table also. We can do this using the interface shown in figure 8.1(d).

6. After selecting a table we can edit (add new filed, delete, edit a filed etc) it. The interface is shown in figure 8.1(e).

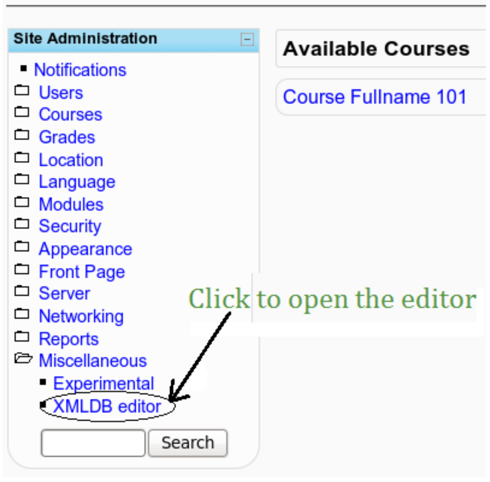
7. Now if we select to add a new field or edit a existing one we will get the interface shown in figure 8.1(f).

8. To update the database we have to get the PHP code. We can get it by clicking on the link *View PHP Code* as shown in figure 8.2(a).

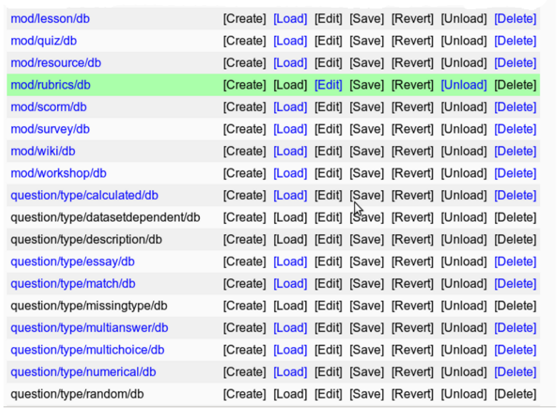
9. After the step 8, we have to copy-paste the code shown in figure 8.2(b) in the update.php file.



(a) Click on Miscellaneous link

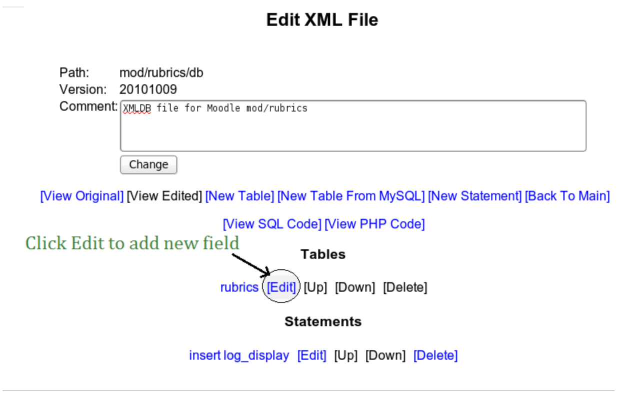


(b) Then select the XMLDB editor



mod/lesson/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/quiz/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/resource/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/rubrics/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/scorm/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/survey/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/wiki/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
mod/workshop/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/calculated/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/datasetdependent/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/description/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/essay/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/match/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/missingtype/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/multianswer/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/multichoice/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/numerical/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]
question/type/random/db	[Create] [Load] [Edit] [Save] [Revert] [Unload] [Delete]

(c) Now load the database needed to be edited



Edit XML File

Path:	mod/rubrics/db
Version:	20101009
Comment:	XMLDB file for Moodle mod/rubrics

Change
[View Original] [View Edited] [New Table] [New Table From MySQL] [New Statement] [Back To Main]
[View SQL Code] [View PHP Code]
Click Edit to add new field

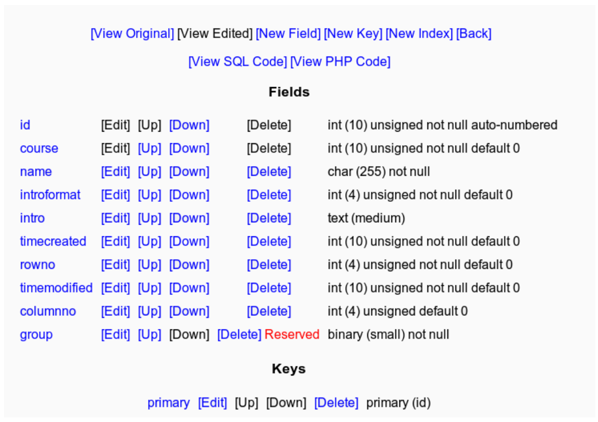
Tables

rubrics [Edit] [Up] [Down] [Delete]

Statements

insert log_display [Edit] [Up] [Down] [Delete]

(d) Select the particular database table



[View Original] [View Edited] [New Field] [New Key] [New Index] [Back]
[View SQL Code] [View PHP Code]

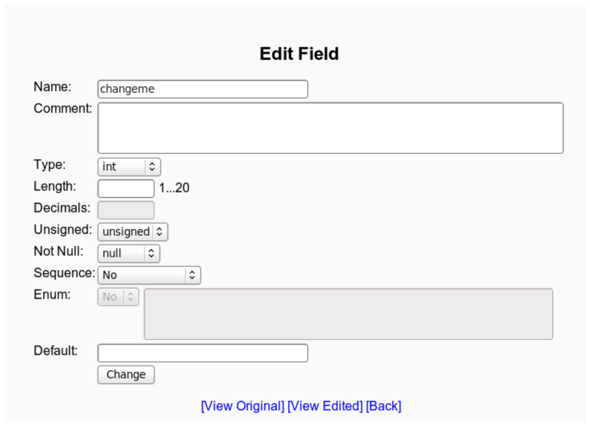
Fields

id	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null auto-numbered
course	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
name	[Edit]	[Up]	[Down]	[Delete]	char (255) not null
introformat	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned not null default 0
intro	[Edit]	[Up]	[Down]	[Delete]	text (medium)
timecreated	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
rowno	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned not null default 0
timemodified	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
columnno	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned default 0
group	[Edit]	[Up]	[Down]	[Delete] Reserved	binary (small) not null

Keys

primary [Edit] [Up] [Down] [Delete] primary (id)

(e) Edit the selected table using this interface

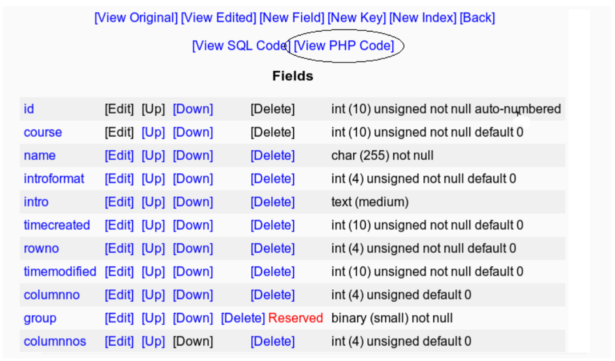


Edit Field

Name:	changeme
Comment:	
Type:	int
Length:	1...20
Decimals:	
Unsigned:	unsigned
Not Null:	null
Sequence:	No
Enum:	No
Default:	

(f) Graphic interface to edit a particular field in the table

Figure 8.1: Use of XMLDB editor to edit Moodle database


[View Original] [View Edited] [New Field] [New Key] [New Index] [Back]
[View SQL Code] [View PHP Code]

Fields

id	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null auto-numbered
course	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
name	[Edit]	[Up]	[Down]	[Delete]	char (255) not null
introformat	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned not null default 0
intro	[Edit]	[Up]	[Down]	[Delete]	text (medium)
timecreated	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
rowno	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned not null default 0
timemodified	[Edit]	[Up]	[Down]	[Delete]	int (10) unsigned not null default 0
columnno	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned default 0
group	[Edit]	[Up]	[Down]	[Delete] Reserved	binary (small) not null
columnnos	[Edit]	[Up]	[Down]	[Delete]	int (4) unsigned default 0

(a) Click on view PHP code to get the PHP code to update the database

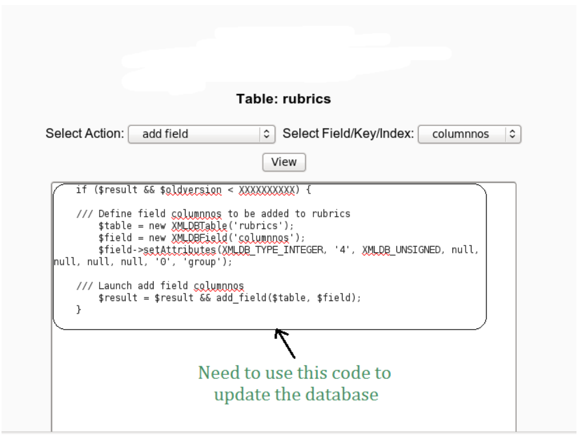


Table: rubrics
Select Action: add field
Select Field/Key/Index: columnnos
View
Need to use this code to update the database

```
    if ($result && $oldversion < XXXXXXXXXX) {

    /// Define field columnnos to be added to rubrics
        $table = new XMLDBTable('rubrics');
        $field = new XMLDBField('columnnos');
        $field->setAttributes(XMLDB_TYPE_INTEGER, '4', XMLDB_UNSIGNED, null,
null, null, null, '0', 'group');

    /// Launch add field columnnos
        $result = $result && add_field($table, $field);
    }
```

(b) Now we need to copy-paste this code in update.php file of the module

Figure 8.2: Use of XMLDB editor to edit Moodle database

8.2 Database Structure of Moodle

Moodle database is well structured. After installation, the 1.9.10 version of Moodle will create about 210 database tables. We will not discuss all the database structure of Moodle. But before creating an activity module it is important to know about some tables of Moodle database.

- Different information regarding the courses are stored *mdl_course* table.
- For each module there is one table named *mdl_modulename* which stores the information regarding a particular module. It stores information like, the instance number of a module, name, introduction etc.
- There is one table called *mdl_course_modules* which stores information about all the modules we can add in a Moodle courses. The value of the id field of a record in this table is unique across different courses. So Moodle use this id to uniquely identify a particular course module.
- There are two tables named *mdl_blocks* and *mdl_modules* which maps the block names and module names to number(ID).
- The *mdl_groups* stores the group names, id of the groups and the course number to which the groups belong. The *mdl_groups_members* store which user is member of which group.
- Different role information in Moodle is store in the *mdl_roles* table. Role capabilities are stored in *mdl_role_capabilities* table.
- Information regarding the users, like user name, id, passwords are stored in the *mdl_user* table.
- Log of the activity in Moodle also stored in the database, in the *mdl_log* table.

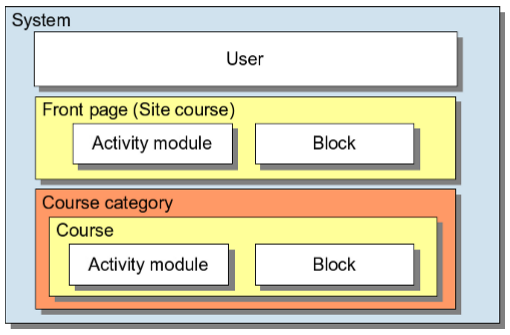


Figure 8.3: Moodle Contexts[26]

8.3 Access Control

Moodle control the access of a user depending upon the ***role*** he has been assigned. In a particular *context*, each user is assigned a particular role in Moodle. Moodle has different types of context. These are

- System (no parent)
- Front page (parent = system)
- Course category (parent = parent category or system)
- Course (parent = category or system)
- Module (parent = course or system)
- Block (parent = course or system)
- User (parent = system)

In Moodle context has certain hierarchy as shown in Figure 8.3. Capabilities assigned at a lower context level will always override a higher context level. For example if a user is assigned the role of *Teacher* for a particular forum (here context level is *Module*) and he is assigned the role of *Student* in the context of course, then in that particular forum *Student* role permissions of the user will be overwritten by the role permissions of *Teacher*.

Moodle comes with 7 predefined roles. These are,

- Administrator
- Course creator
- Teacher
- Non-editing teacher

- Student
- Guest
- Authenticated user

Customize roles can be created in Moodle by the Administrator. Administrator can create the customized role, using the interface provided by the link below.

- *$CFG→wwwroot/admin/roles/manage.php?action=add*

To assign different permissions for different role, in a particular activity module we need to define capabilities in the local */db/access.php* file. Once we defined the capabilities, we can control access by calling the *has_capability ($capability, $contextid, $kill)* function.

We can get the *$contextid* by the function call *get_context_instance(CONTEXT_OPTION, $cm→id)*. Here option can be any of the context defined previously. *$cm* is the course object and *$cm→id* is the id of the course.

How to define the capabilities in the local *access.php* file, can be explained using one example. For the PBL activity module we define the capability *add_discussion* in the following way.

```
    'mod/pbl:add_discussion' => array(
      'captype' => 'write',
      'contextlevel' => CONTEXT_MODULE,
      'legacy' => array(
        'teacher' => CAP_ALLOW,
        'editingteacher' => CAP_ALLOW,
        'admin' => CAP_ALLOW
        )
      )
```

Here *captype* can be either *read* or *write*, *contextlevel* is the level of context for which we want to define the capability. For example *CONTEXT_MODULE* or *CONTEXT_COURSE*. There are four types of permission[32] we can give to a particular role. These are

- *CAP_ALLOW*: If this the permission of a particular role, then *has_capability()* function will return true.
- *CAP_PREVENT*: With this permission, users of a particular role will not have the capability, even though, users of the same role were allowed this capability in a higher context.
- *CAP_PROHIBIT*: If the permission for a particular role is *CAP_PROHIBIT*, then permission of the users of this particular role is completely prohibited and can NOT be overridden at any lower context.

- *CAP_INHERIT*: When the permission is *CAP_INHERIT*, the capability will be inherited from the higher context.

If we change the permission the it only takes effect after we increase the version number of the module and after the next login from the user. Developer can define string for a capability in the file /lang/en_utf8/*modulename*.php file. If no permission is defined, then the capability permission is inherited from a context that is higher than the current context. We can use *require_capability()* function instead of *has_capability()* function. It returns true only if both the combining *has_capability()* with *require_course_login()* returns true.

8.4 Development of a new Activity module

In Moodle teachers can add activity modules inside a course(as shown in figure 8.4). Using activity modules teachers can create different tasks and students need to carry-out those tasks. The most common way to extend the functionalities of Moodle as learning environment is to create a new activity module. Now to create a new activity module we have to follow some steps.

- First of all we need to install Moodle in the machine, where we want to start developing the new module.

- Now let the module name is rubrics. Then a directory named *rubrics* need to be created inside *$HOME/mod* directory. Here $HOME is the home folder where Moodle software package has been installed. Inside this *rubrics* directory we have to create the directory structure as shown below figure 8.5. There is a template named NEWMODULE already given in Moodle web site, using which we can start building the new module. Before we start adding code to the template, we need to replace all the file name from *NEWMODULE* to *rubrics* and all the word named *NEWMODULE* inside the files with *rubrics*.

- After the previous step done, if we go to the administration notification page, it will ask for the permission of the user to made necessary changes to install the new module rubric.

- We need to design the database structure of the module. After designing the database structure, the database needs to be created using the XMLDB editor.

- Now again we have to go to the administration notification page to make the necessary changes in the database structure.

- Table 8.4 lists all the necessary files which are needed to create a new activity module in Moodle and the purpose of each file.

- All the function we write for this specific module should be written in the local *lib.php* file.

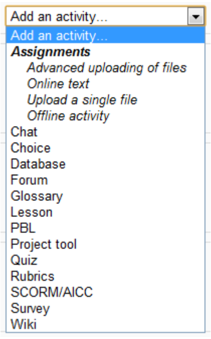


Figure 8.4: Add an activity inside course

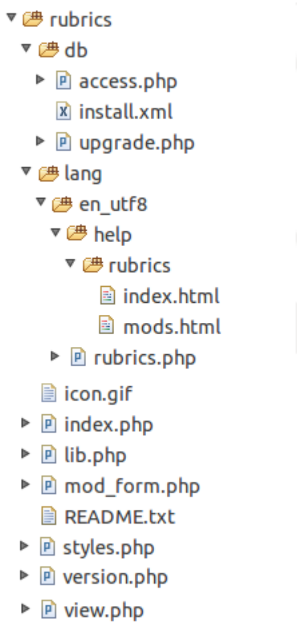

```
rubrics
  db
    access.php
    install.xml
    upgrade.php
  lang
    en_utf8
      help
        rubrics
          index.html
          mods.html
      rubrics.php
  icon.gif
  index.php
  lib.php
  mod_form.php
  README.txt
  styles.php
  version.php
  view.php
```

Figure 8.5: Directory structure for an activity module in Moodle

- While inside a course if we try to add the newly created module, then we will be redirected to *$HOME/course/modedit.php* file. Now this page shows a form which is defined in the local *mod_form.php* file. So according to the need of our module we need to customize this form in the local *mod_form.php* file.

- *view.php* is the homepage of this particular module.

- Before adding any functionalities in the *view.php* file or some other file, we should write the access control capabilities in the local *db/access.php* file, then use these capabilities to restrict the access of different roles.

- Every time we update the structure of the database we need to update the version number in the *version.php* file and write the update function in the *update.php*.

Rubrics Module	
File Name	**Purpose**
index.php	List all instances of the functionality the module provides in a course.
lib.php	This file contains all the functions which are needed to integrate the module with Moodle and all the other functions which are required to implement the module logic.
mod_form.php	The structure of the form which an instructor gets when he select to add a new activity of a particular module, is defined in this file. The same page is displayed when teacher selects to edit the activity.
styles.php	CSS specific to the rubric module is defined in this file
version.php	This file contains the current version number of the module. Whenever we make changes to database we need to update the version number of the module in this file
view.php	When a user select an already created activity, he is directed to this page.
icon.gif	This is the icon of the module
db/access.php	This file contain all the capability definition. This capabilities will be used to restrict the access of the users.
db/install.xml	In this file all the database tables are defined in xml format
db/update.php	Whenever we need to update the database we need to paste proper php code generated by Moodle XMLDB editor in this file.
lang/en_ut8/help/rubrics.php	Some variables which we use in the new module is defined in this file
lang/en_ut8/help/rubrics/index.html	This lists all the help files for the module
lang/en_ut8/help/rubrics/mods.html	The main functionality of the module is described in this file

Table 8.1: **Purpose of different files in an activity module in Moodle**

Chapter 9

Rubric Module

Rubric is an assessment tool. Rubrics are generally used to assess quality or quantity of work, behavior or learning. Though rubrics is an widely used tool, there is no activity module available for easy creation and use of rubric inside Moodle. For the PBL module, rubric is an important part of assessment. So, we have developed a rubric activity module for Moodle. In this chapter we will describe, how this rubric module works and how we have developed the module.

9.1 User Documentation

A standard rubric contain three type of elements, *Criteria*, *Rating scale* and *Rating scale definition*. ***Criteria*** are assessed through the rubric. Different ***rating scale*** are used to rate a particular criteria and for each scale there will be a systematic guideline i.e. ***rating scale definition***. In a good rubric, rating levels should be comprehensive and distinct. In table 9.1 we show an example rubric. Where there are three *criteria* we are assessing and for each criteria four *rating scales* has been defined.

Using the ***Rubric module*** user can create three different types of rubric. These are,

- *Peer evaluation*: These type of rubrics are used to assess the peer students.
- *Self evaluation*: These type of rubrics are used for self assessment.
- *Questionnaire*: These types of rubrics can be used to create questionnaire and view the response.

Now for each type of rubric, in broad scene there are three stages of usability. *Creation of rubric*, *Submission of the rubric* and *Viewing the result*.

- In the creation of rubric stage *teacher* need to choose the type, number of rows and number of columns of the rubric as shown in Figure 9.1. Then fill up the rubric form shown in Figure 9.2.

Rubric of Rubric				
Criteria	**1 Unacceptable**	**2 Acceptable**	**3 Good/Solid**	**4 Exemplary**
Clarity of criteria	Criteria being assessed are unclear, inappropriate and/or have significant overlap	Criteria being assessed can be identified, but are not clearly differentiated	Criteria being assessed are clear, appropriate and distinct	Each criteria is distinct, clearly delineated and fully appropriate for the assignment(s)/course
Distinction between Levels	Little/no distinction can be made between levels of achievement	Some distinction between levels is made, but is not totally clear	Distinction between levels is apparent	Each level is distinct and progresses in a clear and logical order
Reliability of Scoring	Cross-scoring among faculty and/or students often results in significant differences	Cross-scoring by faculty and/or students occasionally produces inconsistent results	There is general agreement between different scorers when using the rubric	Cross-scoring of assignments using rubric results in consistent agreement among scorers

Table 9.1: **Example Rubric[33]**

- After the *teacher* created the rubrics, user can submit the rubric (Figure 9.3). If the rubric is of type ***peer evaluation***, then while submitting user have to choose for whom (destination user) he wants to submit the rubrics. In the case of ***self evaluation*** and ***Questionnaire*** user don't need to select a destination user.

- Now for ***peer evaluation*** and ***self evaluation*** type of rubrics, to see the result user need to select a particular user for whom he wants to see the result. In case of ***Questionnaire*** there are two options to view the result. First one is *overall response*, where user can view the consolidated response from all the users. Other one is *Individual Response* where user can select a particular individual to view his response.

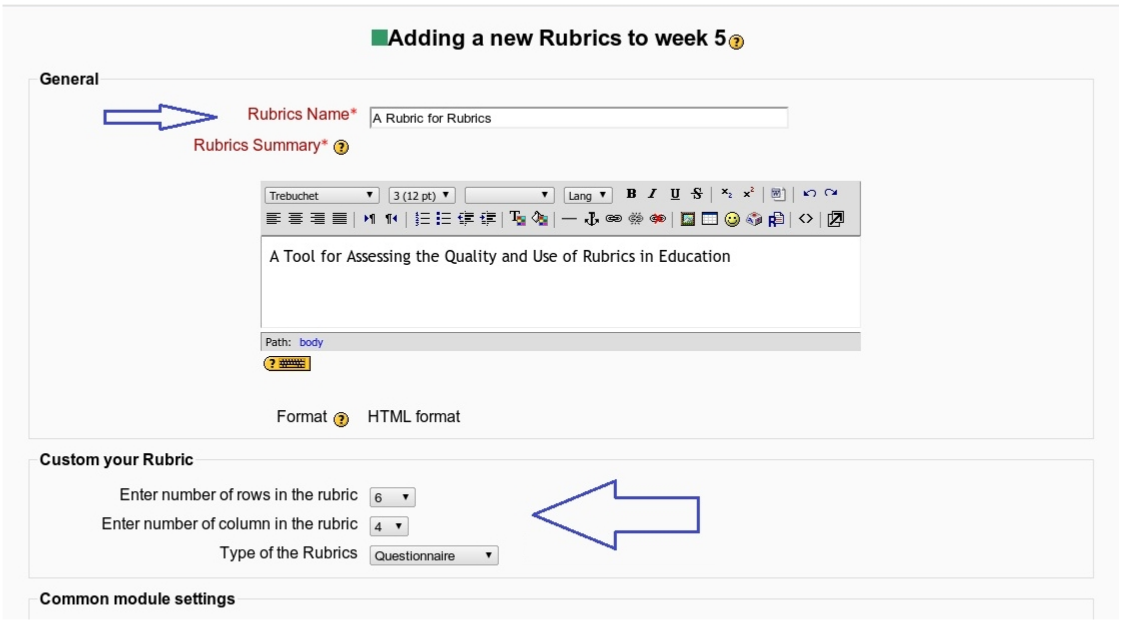
The process flow and dependency is shown in figure 9.6.

9.2 Database Structure

There are four database tables in the Rubric Module. These are

- **mdl_rubrics** : It stores the information regarding the rubric, like name, introduction. It also stores settings of the rubric like type of the rubric, numbers of row and numbers of column etc.

- **mdl_rubrics_form** : This table store the information about the form of a particular rubric. Like name of the *criteria*, different *rating levels* and *rating level definitions*.


Adding a new Rubrics to week 5

General

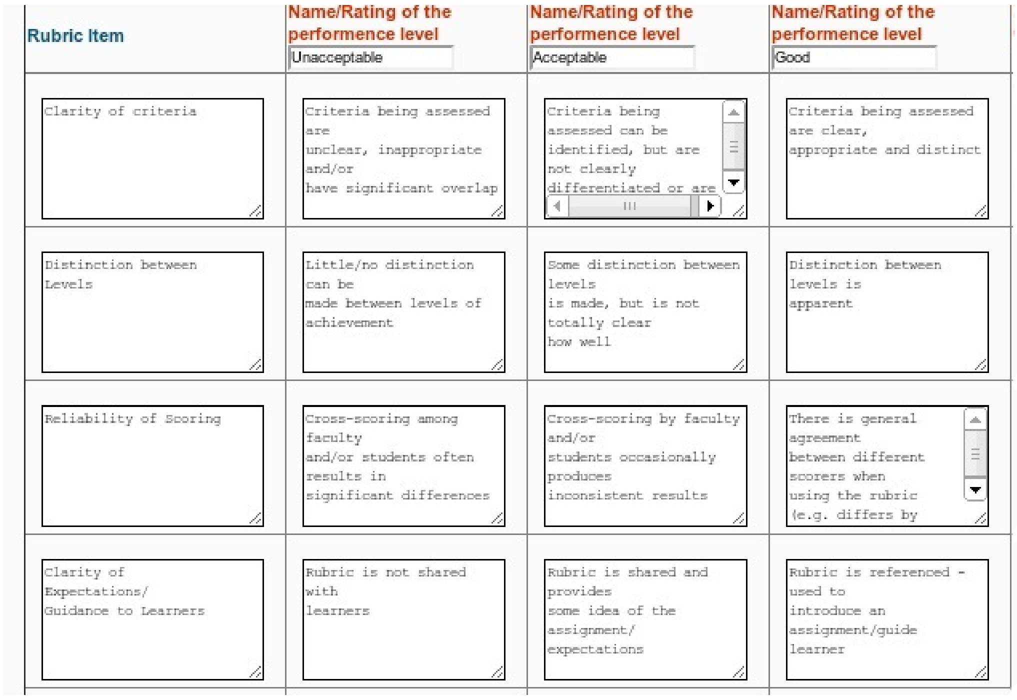
Rubrics Name*	A Rubric for Rubrics
Rubrics Summary*	A Tool for Assessing the Quality and Use of Rubrics in Education
Format	HTML format

Custom your Rubric

Enter number of rows in the rubric	6
Enter number of column in the rubric	4
Type of the Rubrics	Questionnaire

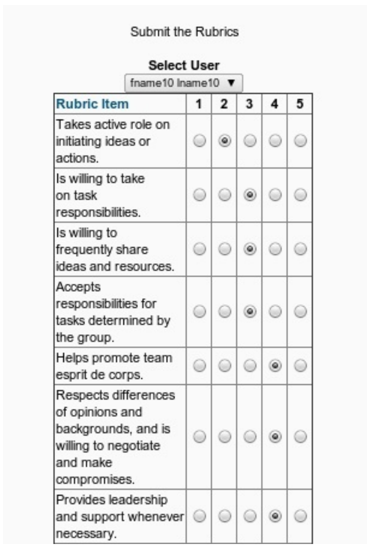
Common module settings

Figure 9.1: Rubric details and different customization options



Rubric Item	Name/Rating of the performence level Unacceptable	Name/Rating of the performence level Acceptable	Name/Rating of the performence level Good
Clarity of criteria	Criteria being assessed are unclear, inappropriate and/or have significant overlap	Criteria being assessed can be identified, but are not clearly differentiated or are	Criteria being assessed are clear, appropriate and distinct
Distinction between Levels	Little/no distinction can be made between levels of achievement	Some distinction between levels is made, but is not totally clear how well	Distinction between levels is apparent
Reliability of Scoring	Cross-scoring among faculty and/or students often results in significant differences	Cross-scoring by faculty and/or students occasionally produces inconsistent results	There is general agreement between different scorers when using the rubric (e.g. differs by
Clarity of Expectations/ Guidance to Learners	Rubric is not shared with learners	Rubric is shared and provides some idea of the assignment/ expectations	Rubric is referenced - used to introduce an assignment/guide learner

Figure 9.2: Rubric form



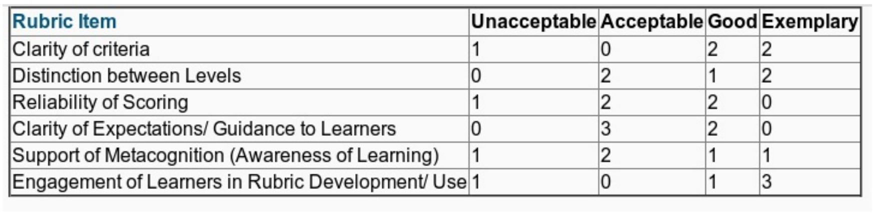
Submit the Rubrics

Select User

fname10 lname10 ▼

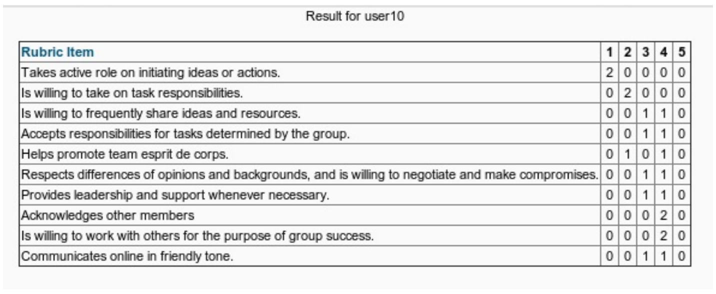
Rubric Item	1	2	3	4	5
Takes active role on initiating ideas or actions.	○	●	○	○	○
Is willing to take on task responsibilities.	○	○	●	○	○
Is willing to frequently share ideas and resources.	○	○	●	○	○
Accepts responsibilities for tasks determined by the group.	○	○	●	○	○
Helps promote team esprit de corps.	○	○	○	●	○
Respects differences of opinions and backgrounds, and is willing to negotiate and make compromises.	○	○	○	●	○
Provides leadership and support whenever necessary.	○	○	○	●	○

Figure 9.3: Rubric submission form



Rubric Item	Unacceptable	Acceptable	Good	Exemplary
Clarity of criteria	1	0	2	2
Distinction between Levels	0	2	1	2
Reliability of Scoring	1	2	2	0
Clarity of Expectations/ Guidance to Learners	0	3	2	0
Support of Metacognition (Awareness of Learning)	1	2	1	1
Engagement of Learners in Rubric Development/ Use	1	0	1	3

Figure 9.4: Overall response



Result for user10

Rubric Item	1	2	3	4	5
Takes active role on initiating ideas or actions.	2	0	0	0	0
Is willing to take on task responsibilities.	0	2	0	0	0
Is willing to frequently share ideas and resources.	0	0	1	1	0
Accepts responsibilities for tasks determined by the group.	0	0	1	1	0
Helps promote team esprit de corps.	0	1	0	1	0
Respects differences of opinions and backgrounds, and is willing to negotiate and make compromises.	0	0	1	1	0
Provides leadership and support whenever necessary.	0	0	1	1	0
Acknowledges other members	0	0	0	2	0
Is willing to work with others for the purpose of group success.	0	0	0	2	0
Communicates online in friendly tone.	0	0	1	1	0

Figure 9.5: Individual response

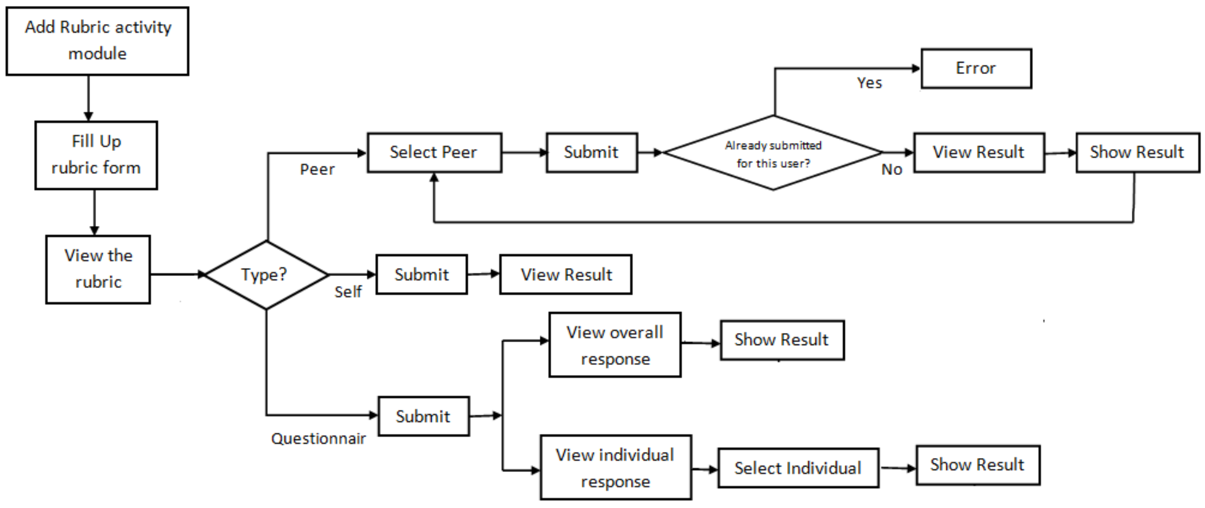


Figure 9.6: Dependency and Process Flow

- **mdl_rubrics_user_record** : If a user submit a rubric, records are stored in this particular table.
- **mdl_rubric_record** : This table is to check if a user has submitted a particular rubric or not.

mdl_rubrics		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
course	bigint(10)	ID of the course
name	varchar(255)	Name of the rubric
introformat	smallint(4)	Introduction format of the rubric
intro	mediumtext	Introduction of the rubric
timecreated	bigint(10)	Creation time of the rubric
rowno	smallint(4)	Number of rows in the rubric
timemodified	bigint(10)	Last modification time of the rubric
columnno	smallint(4)	Number of column
type	mediumtext	Type of the rubric (self/peer/question)

Table 9.2: **Structure of mdl_rubrics table**

9.3 Development Documentation

Development of a rubric module has been done in a modular layered approach. Which is shown in Figure 9.7. As Moodle is used by students, there is always chance of plagiarism, so access

mdl_rubrics_form		
Column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
value	mediumtext	This field will contain the text for all the items in the rubrics form
name	mediumtext	Name of the particular filed in the rubrics form
courseid	bigint(10)	ID of the course
columnid	bigint(10)	Column number
rrowid	bigint(10)	Row number
moduleid	bigint(10)	ID of the rubrics module

Table 9.3: **Structure of mdl_rubrics_form table**

mdl_rubric_record		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
texts	mediumtext	It contains both the source & destination username
moduleid	bigint(10)	It is the module id of the rubrics

Table 9.4: **Structure of mdl_rubric_record table**

mdl_rubrics_user_record		
Column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
rubrics_id	varchar(255)	Id of the rubrics module
row_name	mediumtext	Name of the criteria in a particular row
value	text	Name/Rating for the criteria
src_user	varchar(255)	Source user name
dst_user	varchar(255)	Destination User name
courseid	bigint(10)	ID of the course

Table 9.5: **Structure of mdl_rubrics_user_record table**

control is a very important issue. User should be able to view some thing or access database only after authorization checking. So for this module ***authorization layer*** is the outer most layers. Now, we should not give the user to create a logically wrong database, or to delete some data without proper verification of logic. So ***logic layer*** sits in between the ***authorization*** and ***database layer***. This approach is not applicable always, sometimes we can only determine the functional logic after accessing database. But, if not required database should only be accessed after checking the functional logic.

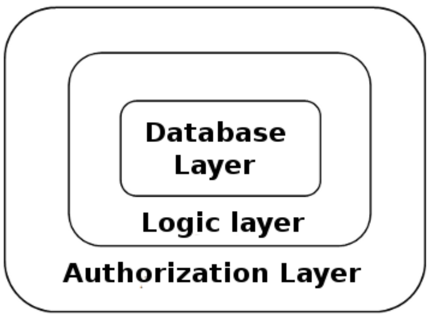


Figure 9.7: Different development layers

9.3.1 Authorization

In rubrics module we check for authorization in two different ways.

- Using already defined API by Moodle
- Using the access control function in local access.php file

To view any of the file in the rubric module user need to be logged in. To check if the user is logged in or not, we use the *require_login()* function defined in the *lib/moodlelib.php* file.

- ***require_login($courseorid, $autologinguest, $cm, $setwantsurltome)***
  - *$courseorid*: It is the course module or id of the current course
  - *$autologinguest*: If it is set, and user is not logged in then he will be logged in as guest
  - *$cm*: Course module object
  - *$setwantsurltome*: True, if we want to set the *$SESSION→wantsurl* variable.

This *require_login()* function checks whether the current user is logged in and is allowed to be in the particular course to view this particular course module. If user is not logged in then he will be redirected to the Moodle *log-in* page. If *$courseorid* is given and the user is not enrolled in that particular course then he will be redirected to the course enrolment page. If *$cm* is given and the module is hidden, then it will not be shown in the course homepage if the current user role is not *teacher*.

Now the local access control capabilities are defined in the local */db/access.php* file. For this rubric module we have defined five capabilities. These are,

1. **viewrubric**: It is to authorize user to view the rubric.
2. **editrubric**: It is to authorize user to edit the rubric.
3. **viewoptions**: It is to authorize user to view different options in rubric homepage.

4. **viewresults**: It is to authorize user to view the results.

5. **viewselfresponse**: It is to authorize user to view self response.

To check for the capability we have used the ***(has_capability('capability name', $context)*** function.

We can also check the role of currently logged in user using the already defined functions by Moodle. For example ***isteacher()*** function is to determine if the currently logged in teacher is a teacher or not, ***isstudent()*** checks if the user is a student or not. This functions are defined in *$HOME/lib/deprecatedlib.php* file.

9.3.2 Functional Logic

The functional logic can be described with respect to the different stages of rubric. There are three stages regarding the use of rubric. These are *creation of rubric*, *submission of the rubric* and *view the result*.

Creation of Rubric:

In the course homepage, if a user selects to add a *rubric* from the *Add an activity* (Figure 8.4) menu, then we will be redirected to the page *$HOME/course/modedit.php*. In this page we will see a form, which is coded in the local *mod_form.php* file. In this file we declare a class *mod_rubrics_mod_form* which extends the class *moodleform_mod*. The class *moodleform_mod* is defined in the *$HOME/course/moodleform_mod.php* file. For easy creation, processing and validation Moodle use Quickform, which is extension of HTML form. In the *mod_rubrics_mod_form* class we define five important element about a particular rubric instance.

1. **name**: It is of type *text*. It will be saved as the name of the rubric.

2. **intro**: It is of type *htmleditor*. It will be saved as the introduction of the rubric in HTML format.

3. **rowno**: It is of type *select*. User can choose a value from 1 to 100, using this select drop-down box. It will be saved as the number of rows in the rubric.

4. **columnno**: It is of type *select*. User can choose a value from 1 to 10, using this select drop-down box. It will be saved as the number of columns in the rubric.

5. **type**: It is of type *select*. It will be saved as the type(self/peer/question) of the rubric.

To create, update or delete a rubric all the functions are defined in the local *lib.php* file. These function are,

- ***rubrics_add_instance($rubrics)***: Here $rubric is the object from the form in *mod_form.php*.

- ***rubrics_update_instance($rubrics)***: Here $rubric is a object.
- ***rubrics_delete_instance($id)***: Here $id is the id of the rubric, which we want to delete.

Now after creation of a new instance of rubric module, user can fill up the rubric form (Figure 9.2) to create the rubric. This rubric form is created according to the number of rows and columns given by the user, while creating the instance of rubric module. Now if the user saves the form, all the data regarding the form will be stored in the *mdl_rubrics_form* table in the database. Each element of the form will be any of these three types,

1. *level_definition*
2. *criteria_name*
3. *rating*

We can identify each item in the rubric form uniquely by the course ID, module ID, row number, column number and criteria name.

Submission of Rubric:

In this phase, depending on the type(self/peer/question) of the rubric we need to show either only the rubric form, or the list of the user along with the rubric form. To show the list of users(only of role 'student'), we used the functions described below.

- ***get_role_users($student_role→id, $context)***: Here *$student_role→id* is the id of role named *student* as defined in the *mdl_roles* table in Moodle, *$context* is the context of the module.

To show the rubric form, we use another function defined in the local *lib.php* file,

- ***show_rubric_form($courseid,$moduleid)***: Here *$courseid* is the ID of the current course, and *$moduleid* is the ID of the current rubric module.

Now when user submit a rubric, records will be stored in the *rubrics_record* table. Along with the *criteria_name* and *rating*, depending upon the type of the rubric either only the source user-name(questionnaire type of rubric) or both the source user-name and destination user-name(peer and self type rubric) will be stored in the *mdl_rubrics_user_record* table in the database.

View Result:

To view the result homepage we used the function,

- ***show_rubric_result($courseid,$moduleid)***: Here *$courseid* is the ID of the current course, and *$moduleid* is the ID of the current rubric module.

Now if the user wants to check the result for a particular user then from the table *mdl_rubric_record* we check if any record for that particular user is available or not. If available, then from the *mdl_rubrics_user_record* we fetch the record and print it, else we print an error message.

9.3.3 Database Layer

For the rubrics module all the database access are done using the library functions defined by Moodle. The functions which we have used are

- ***get_records($table, $field, $value)***: This function is used to get the records from the table *$table*, where *$filed1* is having the value *$value1* and so on. *$field*s and *$value*s are the optional fields.
- ***get_record($table, $field1, $value1, $field2, $value2, $field3, $value3)***: This functions works the same way as *get_records()*, except it fetches only a single row from the *$table*.
- ***insert_record($table, $dataobject)***: This function is used to insert the object *$dataobject* into the table named *$table*.
- ***update_record($table, $dataobject)***: This function updates the record of the table *$table*, having the same id as of *$dataobject→id*.
- ***count_records($table, $field1, $value1, $field2, $value2, $field3, $value3)***: This function is used to get the number of records matches in the table *$table*, where *$filed1* is having the value *$value1* and so on. For this function, *$field1*, *$value1*, *$field2*, *$value2*, *$field3*, *$value3* are the optional arguments.
- ***delete_records($table, $field1, $value1, $field2, $value2, $field3, $value3)***: This function used to delete records from the table named *$table*. Other argument works same as the previous function.

All these functions are defined in the *$HOME/lib/dmllib.php* file.

9.4 Challenges

The main challenge to develop this particular module is to know the structure of Moodle and different API's of Moodle.

9.4.1 Moodle Structure

The basic installation of Moodle 1.9.x will create almost 6500 files in the local directory. The library files which are used frequently, like *moodlelib.php*, *weblib.php*, *accesslib.php* each having more than 5000 lines of code. Therefore, Moodle is a large software, and initially it takes time to figure out how a particular feature is working internally. Other than this, before developing a module we need to know *basic databases structure of Moodle*, *how access is controlled in Moodle*, *how the XMLDB editor works*, *about the directory structure of a activity module*, *how the Moodle quick-form extension works* etc.

9.4.2 Moodle Form

Moodle use the HTML form extension *Quickform* for easy development, validation and processing of forms. So while coding inside Moodle, if we want to use forms it's better to use the quickform extensio. The form to add a particular instance of rubrics module as shown in Figure 9.1, is populated using the Moodle's quickform extension. But one drawback of this quickform is that input in the tabular format is not directly supported. However while creation of rubric, for the user it would be more user friendly if he can give the input in a table to fill up the rubric form as shown in Figure 9.2. Initially the rubric form was coded using the quickform of Moodle. But as the GUI was not much user friendly, latter we have change the code and use normal HTML form and wrote the validation functions.

9.5 Future Work

There are some modification and extra features need to be added in the rubrics module.

- There should be detailed options for the teacher if they want to show the result to the students or not depending on the type of the rubric.
- There should be an option to classify the rubric into categories depending upon the type of the rating. The type of the rating can be of two types *numeric* and *text*. If the rating is numeric, then in the result we will be able to show different analysis of the ratings.
- Now if teacher is willing to use the rubric as an summative assessment, then the rubric module need to integrate with the grade module of Moodle.
- This module should be compatible with groups. So if the rubric type is Peer then user can only evaluate members of the group.

Chapter 10

PBL Module

The PBL module has been developed to support all the steps (described in section 2.2) of Problem Based Learning in a better way than the existing Moodle. This PBL module will help to execute Problem Based Learning in both distance learning and blended learning scenario. This module has been developed as an *activity module* in Moodle. It works for *Moodle 1.9.X* version. Now in this module, there are different sub-activities we can add to have all the functionalities. These are, ***discussion***, ***RLF***, ***solution***, ***submission***, ***file*** and ***rubric***. *Discussion* can be of three type, *chat*, *wiki*, *forum*. Among these sub-activities of PBL *chat*[28], *wiki*[27], *forum*[29] are the activity module for Moodle and these features comes with default Moodle package. For the submission module we have used the *assignment*[31] module of Moodle. For the *file* sub-activity we have used the a block plug-in named *File-Manager*[30]. The *rubrics* sub-activity in PBL module is also an activity module of Moodle, but it is does not come with the standard package, we have developed it. *RLF* and *submission* are not activity module, they are specifically developed to be used in this PBL module.

10.1 User Documentation

For the PBL module we will discuss the different features with respect to the different steps of PBL. Before using the PBL module, some course level settings are required. Like *global search* need to be enabled and *course level group setting* should be ***separate group***, the course should be of *weekly* type, the *rubric* module and the *File-Manager* block needed to be installed along with the standard Moodle package. Now in section 2.2 we described the different steps of PBL. The first step is *group formation*.

10.1.1 Group Formation

For the PBL module, we use the Moodle's grouping feature. The groups are created at the course level and the same groups are used in the PBL module. So before setting up the PBL module teacher need to create the groups. The group creation interface is shown in figure 10.1.

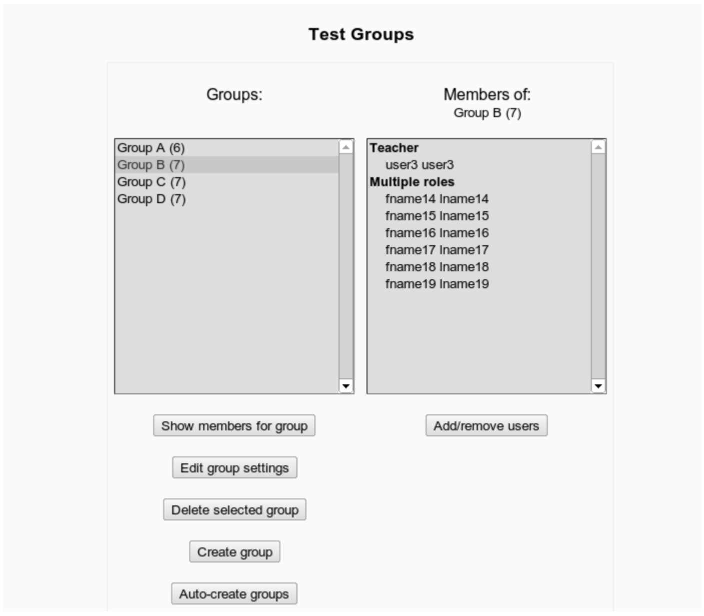


Figure 10.1: Group Creation

10.1.2 Facilitator Allocation

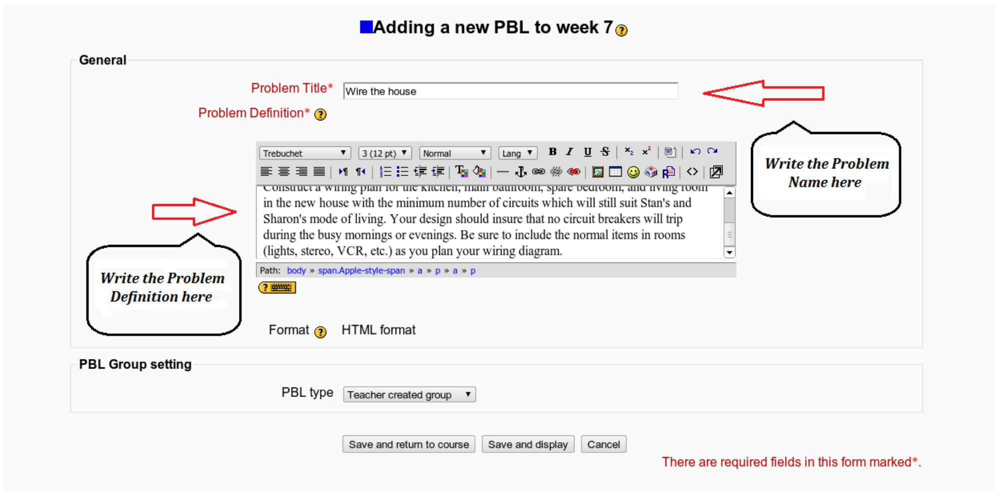
While creating the groups, we can make a particular teacher as the part of the group. Now the teacher who is a part of the group is the facilitator of that particular group. There can be multiple facilitators for a particular group and a facilitator can facilitate multiple groups.

10.1.3 Problem Presentation

After the group creation and facilitator allocation steps are done, teachers can create a new PBL activity in the Moodle course. While adding the PBL activity, teacher will get the option to write the *PBL name*, and the *Problem definition*. Figure 10.2 shows the interface where user can insert different details of PBL activity. In the Figure 10.2 user is creating a PBL activity named *Wiring a home PBL*. This name and problem definition will be shown in the PBL homepage, after user saves it.

10.1.4 Identification of RLF

After the problem presentation done, group members can create a list of *Relevant Learning Facts(RLF)*. User will find a link in the PBL homepage, i.e. *Add RLF* (figure 10.6). If a user clicks on this links he/she will be redirected to a page, where user can save the name of the RLF and description for the RLF. Now after creating the RLF it will be shown in the PBL homepage. If the currently logged-in user has added a particular RLF, then he will get the option to delete it. User can mark a particular RLF as known or unknown, and in the PBL homepage RLFs are shown in two category: *Unknown* and *Known*. For example, let the RLF is *Electrical Circuit*,


Adding a new PBL to week 7

General

Problem Title* Wire the house

Problem Definition*

Construct a wiring plan for the kitchen, main bathroom, spare bedroom, and living room
in the new house with the minimum number of circuits which will still suit Stan's and
Sharon's mode of living. Your design should insure that no circuit breakers will trip
during the busy mornings or evenings. Be sure to include the normal items in rooms
(lights, stereo, VCR, etc.) as you plan your wiring diagram.

Path: body » span.Apple-style-span » a » p » a » p

Format HTML format

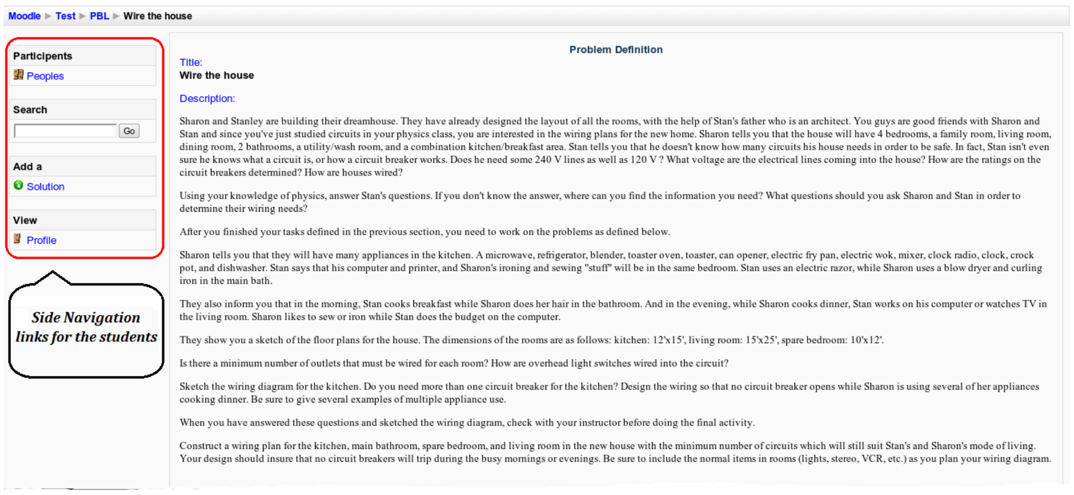
PBL Group setting

PBL type Teacher created group

There are required fields in this form marked*.

Trebuchet
3 (12 pt)
Normal
Lang
Write the Problem Name here
Write the Problem Definition here
Save and return to course
Save and display
Cancel

Figure 10.2: Problem Presentation


Moodle ▸ Test ▸ PBL ▸ Wire the house

Problem Definition

Title:
Wire the house

Description:

Sharon and Stanley are building their dreamhouse. They have already designed the layout of all the rooms, with the help of Stan's father who is an architect. You guys are good friends with Sharon and Stan and since you've just studied circuits in your physics class, you are interested in the wiring plans for the new home. Sharon tells you that the house will have 4 bedrooms, a family room, living room, dining room, 2 bathrooms, a utility/wash room, and a combination kitchen/breakfast area. Stan tells you that he doesn't know how many circuits his house needs in order to be safe. In fact, Stan isn't even sure he knows what a circuit is, or how a circuit breaker works. Does he need some 240 V lines as well as 120 V ? What voltage are the electrical lines coming into the house? How are the ratings on the circuit breakers determined? How are houses wired?

Using your knowledge of physics, answer Stan's questions. If you don't know the answer, where can you find the information you need? What questions should you ask Sharon and Stan in order to determine their wiring needs?

After you finished your tasks defined in the previous section, you need to work on the problems as defined below.

Sharon tells you that they will have many appliances in the kitchen. A microwave, refrigerator, blender, toaster oven, toaster, can opener, electric fry pan, electric wok, mixer, clock radio, clock, crock pot, and dishwasher. Stan says that his computer and printer, and Sharon's ironing and sewing "stuff" will be in the same bedroom. Stan uses an electric razor, while Sharon uses a blow dryer and curling iron in the main bath.

They also inform you that in the morning, Stan cooks breakfast while Sharon does her hair in the bathroom. And in the evening, while Sharon cooks dinner, Stan works on his computer or watches TV in the living room. Sharon likes to sew or iron while Stan does the budget on the computer.

They show you a sketch of the floor plans for the house. The dimensions of the rooms are as follows: kitchen: 12'x15', living room: 15'x25', spare bedroom: 10'x12'.

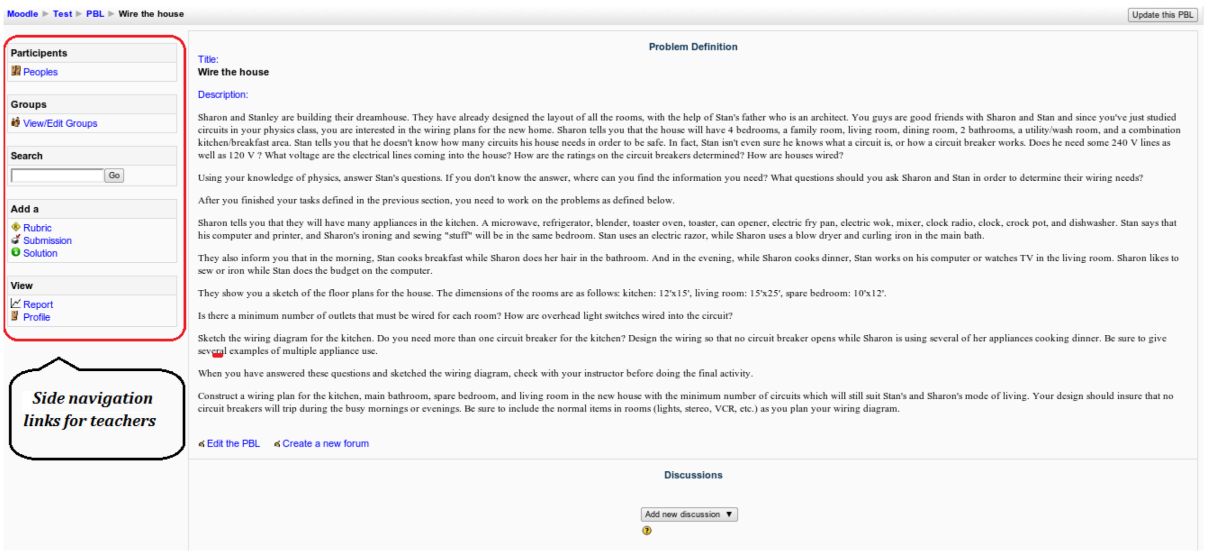
Is there a minimum number of outlets that must be wired for each room? How are overhead light switches wired into the circuit?

Sketch the wiring diagram for the kitchen. Do you need more than one circuit breaker for the kitchen? Design the wiring so that no circuit breaker opens while Sharon is using several of her appliances cooking dinner. Be sure to give several examples of multiple appliance use.

When you have answered these questions and sketched the wiring diagram, check with your instructor before doing the final activity.

Construct a wiring plan for the kitchen, main bathroom, spare bedroom, and living room in the new house with the minimum number of circuits which will still suit Stan's and Sharon's mode of living. Your design should insure that no circuit breakers will trip during the busy mornings or evenings. Be sure to include the normal items in rooms (lights, stereo, VCR, etc.) as you plan your wiring diagram.

Figure 10.3: PBL homepage of the students after problem presentation



Problem Definition

Title:
Wire the house

Description:

Sharon and Stanley are building their dreamhouse. They have already designed the layout of all the rooms, with the help of Stan's father who is an architect. You guys are good friends with Sharon and Stan and since you've just studied circuits in your physics class, you are interested in the wiring plans for the new home. Sharon tells you that the house will have 4 bedrooms, a family room, living room, dining room, 2 bathrooms, a utility/wash room, and a combination kitchen/breakfast area. Stan tells you that he doesn't know how many circuits his house needs in order to be safe. In fact, Stan isn't even sure he knows what a circuit is, or how a circuit breaker works. Does he need some 240 V lines as well as 120 V ? What voltage are the electrical lines coming into the house? How are the ratings on the circuit breakers determined? How are houses wired?

Using your knowledge of physics, answer Stan's questions. If you don't know the answer, where can you find the information you need? What questions should you ask Sharon and Stan in order to determine their wiring needs?

After you finished your tasks defined in the previous section, you need to work on the problems as defined below.

Sharon tells you that they will have many appliances in the kitchen. A microwave, refrigerator, blender, toaster oven, toaster, can opener, electric fry pan, electric wok, mixer, clock radio, clock, crock pot, and dishwasher. Stan says that his computer and printer, and Sharon's ironing and sewing "stuff" will be in the same bedroom. Stan uses an electric razor, while Sharon uses a blow dryer and curling iron in the main bath.

They also inform you that in the morning, Stan cooks breakfast while Sharon does her hair in the bathroom. And in the evening, while Sharon cooks dinner, Stan works on his computer or watches TV in the living room. Sharon likes to sew or iron while Stan does the budget on the computer.

They show you a sketch of the floor plans for the house. The dimensions of the rooms are as follows: kitchen: 12'x15', living room: 15'x25', spare bedroom: 10'x12'.

Is there a minimum number of outlets that must be wired for each room? How are overhead light switches wired into the circuit?

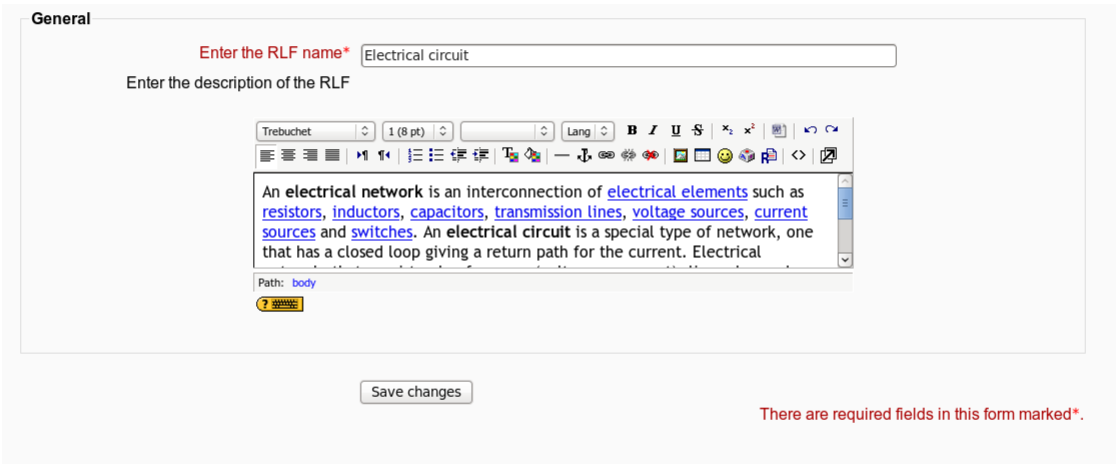
Sketch the wiring diagram for the kitchen. Do you need more than one circuit breaker for the kitchen? Design the wiring so that no circuit breaker opens while Sharon is using several of her appliances cooking dinner. Be sure to give several examples of multiple appliance use.

When you have answered these questions and sketched the wiring diagram, check with your instructor before doing the final activity.

Construct a wiring plan for the kitchen, main bathroom, spare bedroom, and living room in the new house with the minimum number of circuits which will still suit Stan's and Sharon's mode of living. Your design should insure that no circuit breakers will trip during the busy mornings or evenings. Be sure to include the normal items in rooms (lights, stereo, VCR, etc.) as you plan your wiring diagram.

Discussions

Figure 10.4: PBL homepage of the teachers after problem presentation



General

Enter the RLF name* Electrical circuit

Enter the description of the RLF

An **electrical network** is an interconnection of electrical elements such as resistors, inductors, capacitors, transmission lines, voltage sources, current sources and switches. An **electrical circuit** is a special type of network, one that has a closed loop giving a return path for the current. Electrical

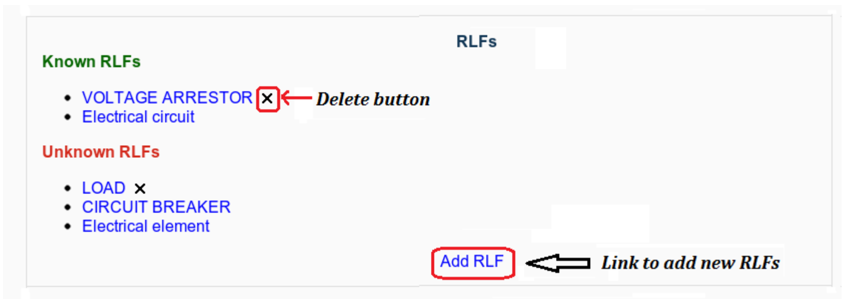
There are required fields in this form marked*.

Figure 10.5: Addition of RLF

then figure 10.5 shows the interface to add RLF . Figure 10.6 shows how RLFs are shown in the PBL homepage as *Known* and *Unknown* category and the link to *Add RLF*. Now After adding the RLF, if we click on the link in the PBL homepage, we can view the name, description of the RLF as shown in figure 10.7.

10.1.5 Discussion

To discuss the doubts regarding the problem definition, there is a forum associated with the problem definition of the PBL. Only teacher has the permission to create and update this forum and after creation, both teacher and student can use the forum. Along with the forum to discuss doubts regarding the problem definition, teacher can any time add three types of discussion(shown in figure 10.8) in the PBL module.


RLFs

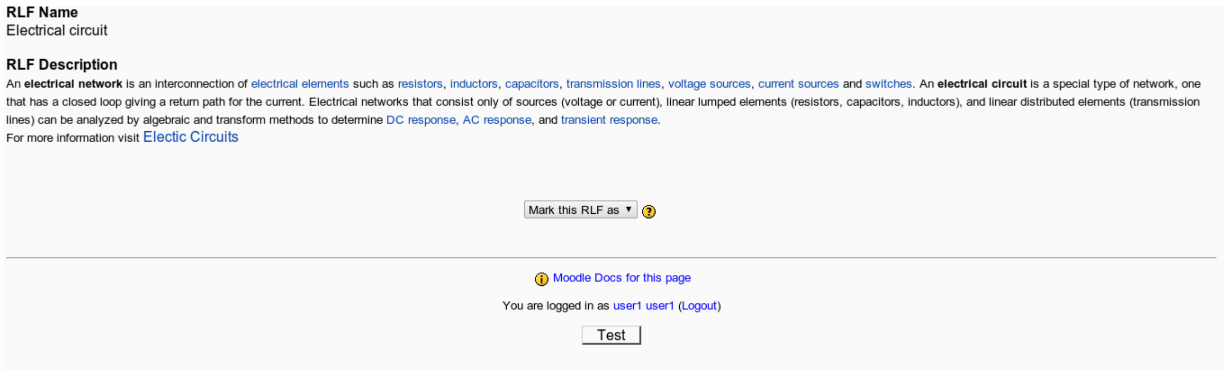
Known RLFs

- VOLTAGE ARRESTOR ✕
- Electrical circuit

Unknown RLFs

- LOAD ✕
- CIRCUIT BREAKER
- Electrical element

Figure 10.6: RLF in PBL homepage


RLF Name
Electrical circuit

RLF Description
An **electrical network** is an interconnection of electrical elements such as resistors, inductors, capacitors, transmission lines, voltage sources, current sources and switches. An **electrical circuit** is a special type of network, one that has a closed loop giving a return path for the current. Electrical networks that consist only of sources (voltage or current), linear lumped elements (resistors, capacitors, inductors), and linear distributed elements (transmission lines) can be analyzed by algebraic and transform methods to determine DC response, AC response, and transient response.
For more information visit Electic Circuits

You are logged in as user1 user1 (Logout)

Figure 10.7: View of the RLF *Electric Circuit*

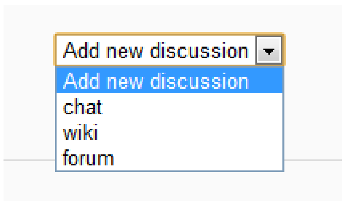


Figure 10.8: Different type of discussions

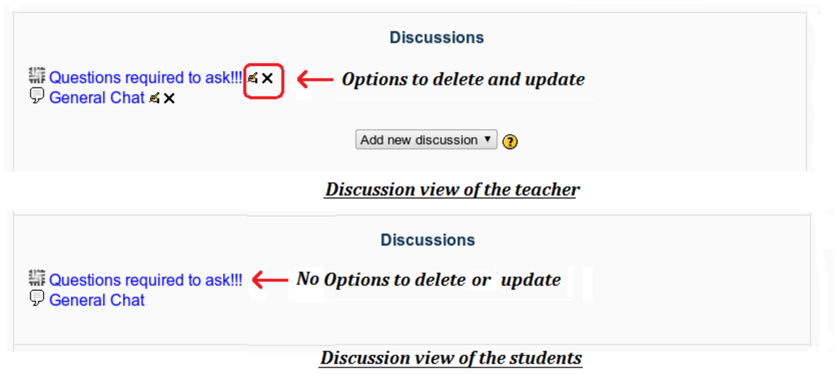

Discussions
Questions required to ask!!!
General Chat
Options to delete and update
Add new discussion
Discussion view of the teacher
Discussions
Questions required to ask!!!
General Chat
No Options to delete or update
Discussion view of the students

Figure 10.9: Teacher's and student's view of the discussions

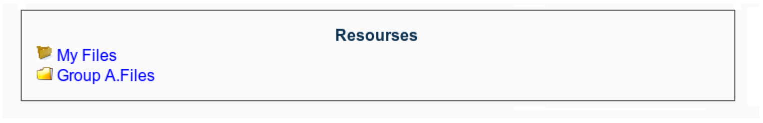


Figure 10.10: File Sharing links

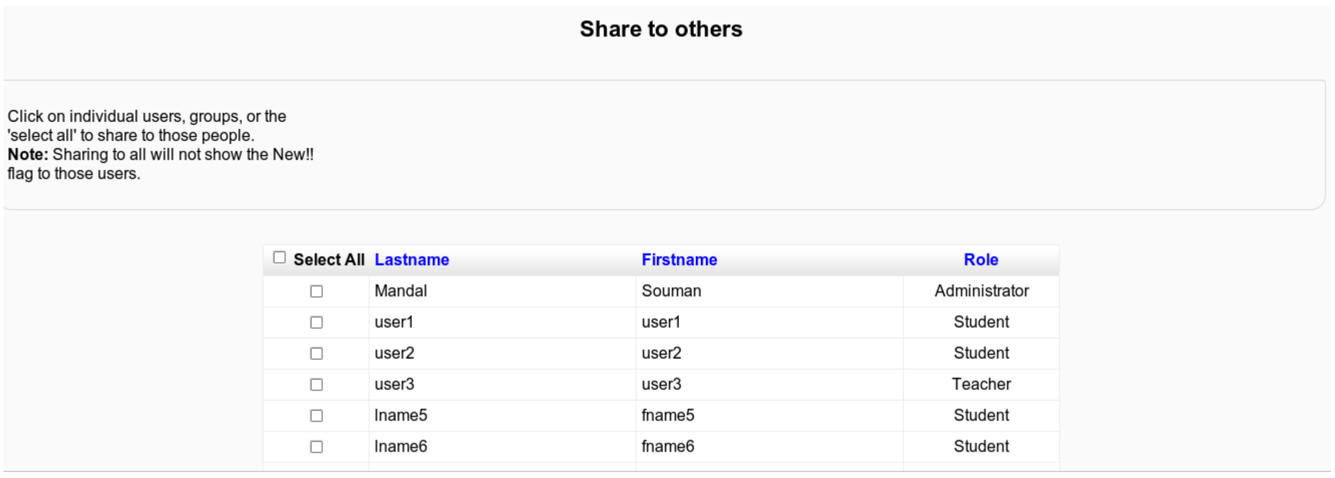
These are

- *Chat*: User can send text messages to the on-line friends.
- *Forum*: Forum can be used as the tool for asynchronous communication.
- *Wiki*: Using the wiki members of group can collaboratively create documents.

After adding a discussion by the teacher, both the teacher and student can use it. Teacher can update and delete discussions. The teacher's view and the student's view of the discussions in the PBL homepage is shown in figure 10.9.

10.1.6 Resource Sharing

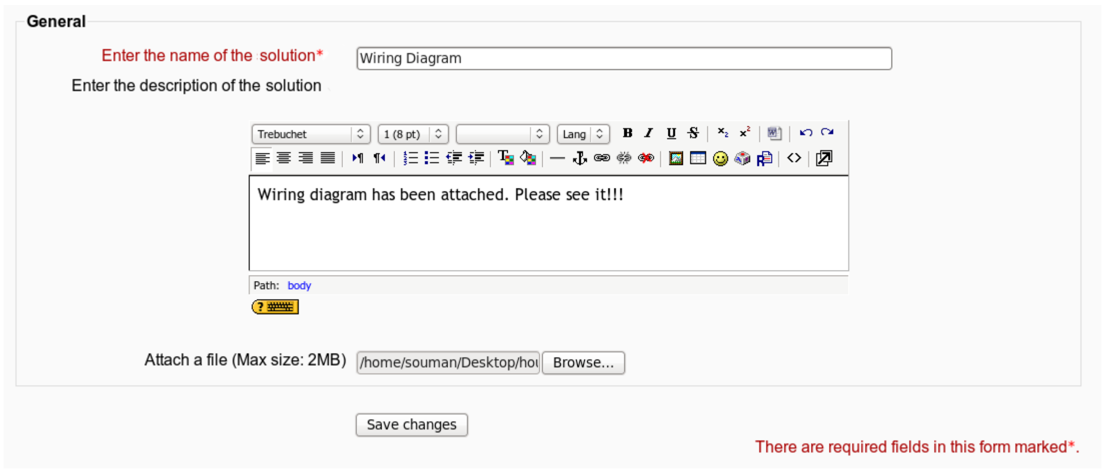
User of the PBL module can share resources by uploading file and share them with others using the *File Manager* block. There are two places (shown in figure 10.10) where user can upload files, first is the **My Files** and the second is the ***Group X* Files**, where *Group X* is the name of the group, in which user is a member. Now if the user upload the files or folders in ***Group X* Files**, then all the members of the group can use those files. But if the files are uploaded in **My Files**, then only the logged-in user can view those files. But user can share any file with anyone else in the course if he wants to, as shown in figure 10.11.


Share to others

Click on individual users, groups, or the 'select all' to share to those people.
Note: Sharing to all will not show the New!! flag to those users.

☐ Select All	Lastname	Firstname	Role
☐	Mandal	Souman	Administrator
☐	user1	user1	Student
☐	user2	user2	Student
☐	user3	user3	Teacher
☐	lname5	fname5	Student
☐	lname6	fname6	Student

Figure 10.11: File Sharing links


General

Enter the name of the solution*	Wiring Diagram
Enter the description of the solution	Wiring diagram has been attached. Please see it!!!
Attach a file (Max size: 2MB)	/home/souman/Desktop/hoi

There are required fields in this form marked*.

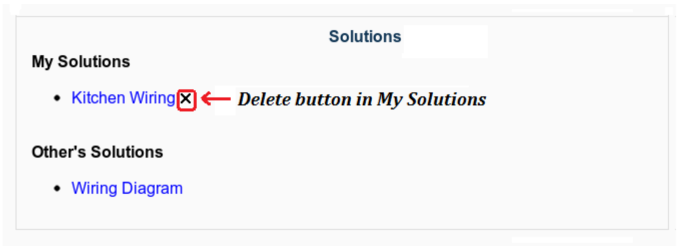
Figure 10.12: Solution Proposal

10.1.7 Solution Proposal

In this PBL module a particular user can propose upto 5 solutions in a PBL activity. To add a solution user needs to click on the side navigation link *solution*. Let for the *Wiring a home* PBL, let the User can wants to propose a solution for the wiring diagram. Then he can click on the solution link, to get the interface as shown in figure 10.12. Here user can give a description of the solution and can attach a file with it. After user submits the proposal, it will be shown in the PBL homepage. Once a user proposes a solution all the members of the group can view it. User can delete his/her proposal at any time.

10.1.8 Solution Submission

Teacher can create the solution submission link in the PBL module. There are four different types of submission possible.


Solutions

My Solutions

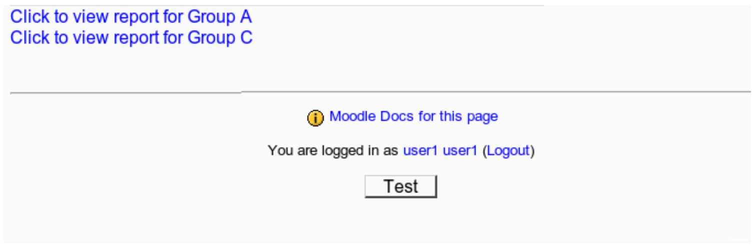
- Kitchen Wiring

Delete button in My Solutions

Other's Solutions

- Wiring Diagram

Figure 10.13: Solutions as shown in the PBL homepage


Click to view report for Group A
Click to view report for Group C

Moodle Docs for this page

You are logged in as user1 user1 (Logout)

Figure 10.14: Link to the report of the groups

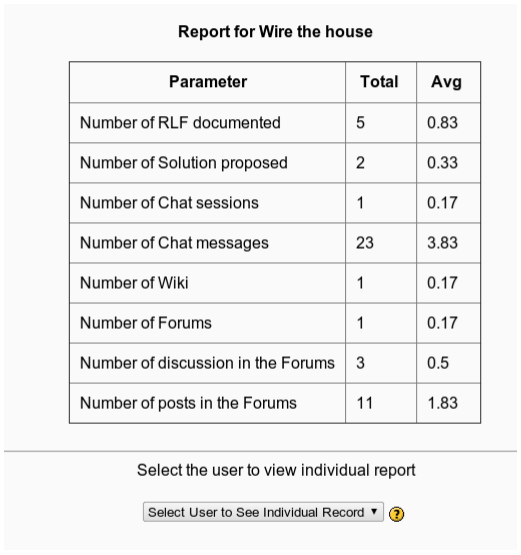
These are

- **Upload a single file**: User can upload a single file.
- **Advanced uploading of files**: User can submit multiple files. It also allows students to type a message alongside their submission and returning a file as feedback.
- **Online text**: Students type the solution directly into Moodle, teachers can provide inline feedback.
- **Offline Activity**: Teachers provide a description and due date for an assignment outside of Moodle. A grade and feedback can be recorded in Moodle.

Students can submit the final solution, after the teacher creates the submission link.

10.1.9 Evaluation

For evaluation of PBL, peer evaluation and self evaluation are important. This can be easily done using the *rubrics* module. Using this module teacher can create different types of rubric. To help with the evaluation process, teacher can view a report for the PBL. There are two options to view the report. These are report of the individual and report of overall group. Teacher will have the links to view the overall report of the groups, he facilitate. In the *Wiring a home* PBL, teacher *user1* facilitate 2 groups, so he can view the report of this two groups, as shown in figure 10.14. Figure 10.15 shows the report of the Group A in the *Wiring a home* PBL. Figure 10.16 shows the report of the *user 3* for the *Wiring a home* PBL in Group A.

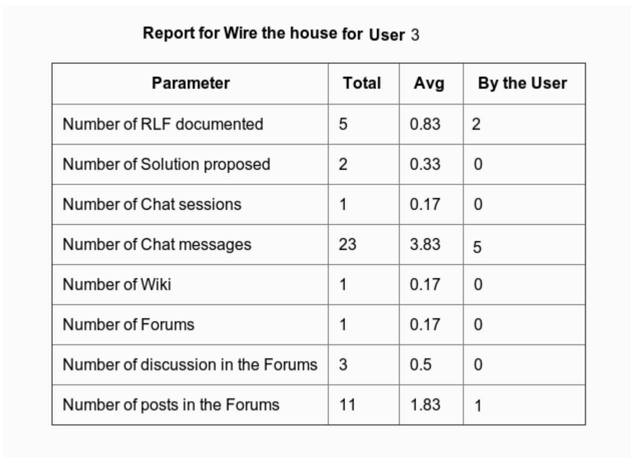


Report for Wire the house

Parameter	Total	Avg
Number of RLF documented	5	0.83
Number of Solution proposed	2	0.33
Number of Chat sessions	1	0.17
Number of Chat messages	23	3.83
Number of Wiki	1	0.17
Number of Forums	1	0.17
Number of discussion in the Forums	3	0.5
Number of posts in the Forums	11	1.83

Select the user to view individual report

Figure 10.15: Report of Group A



Report for Wire the house for User 3

Parameter	Total	Avg	By the User
Number of RLF documented	5	0.83	2
Number of Solution proposed	2	0.33	0
Number of Chat sessions	1	0.17	0
Number of Chat messages	23	3.83	5
Number of Wiki	1	0.17	0
Number of Forums	1	0.17	0
Number of discussion in the Forums	3	0.5	0
Number of posts in the Forums	11	1.83	1

Figure 10.16: Report of User 3

10.1.10 Other features

The other features in the PBL module are *search*, *View profile* and *View participants*.

- Using the *search* feature user can search globally in the course.
- Using *view profile* user can view his/her profile and edit the profile.
- Using the *View participants* link user can view the profile of the other members in the group. There is a option to send message to the other user after user click on the profile of a particular user.

10.2 Database Structure

There are eight different database tables created for the PBL module. These are,

- **mdl_pbl**: This table stores the general information and settings of a PBL modules. General information like PBL name, introduction, creation-time, settings like the type of the PBL are stored.

- **mdl_pbl_discuss**: This table will store the information regarding the different discussion module we add inside the PBL module.

- **mdl_pbl_forum**: It stores the information regarding the forum, created to discuss the doubts related to the problem definition.

- **mdl_pbl_rlf**: Stores the information regarding the RLFs of PBL modules.

- **mdl_pbl_rlf_user**: Stores the information if a user mark the RLF as read.

- **mdl_pbl_solution**: Stores the information regarding the different solution provided by the students.

- **mdl_pbl_groups**: It will store the names of the PBL groups and their ID.

- **mdl_pbl_groups_members**: It will store to which PBL group a particular user belongs to.

mdl_pbl		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
course	bigint(10)	ID of the course
name	varchar(255)	Name of the PBL
intro	mediumtext	Introduction of the PBL
introformat	smallint(4)	Format of introduction
timecreated	bigint(10)	Time-stamp when the PBL module was created
timemodified	bigint(10)	Time-stamp when the PBL module was modified last
pbltype	mediumtext	Type of the PBL

Table 10.1: **Structure of table mdl_pbl**

mdl_pbl_rlf		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
usrid	bigint(20)	Who submitted the RLF
moduleid	bigint(10)	ID of the PBL
courseid	bigint(10)	ID of the course
groupid	bigint(10)	ID of the group, user belongs to
description	mediumtext	Description of the RLF

Table 10.2: **Structure of table mdl_pbl_rlf**

mdl_pbl_rlf_user		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
rlfid	bigint(10)	ID of the RLF
userid	bigint(10)	ID of the user

Table 10.3: **Structure of table mdl_pbl_rlf_user**

mdl_pbl_discuss		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
type	text	Type of different discussion
pblid	bigint(10)	ID of the PBL module
userid	bigint(10)	ID of the user who created it
courseid	bigint(10)	ID of the course
groupid	bigint(10)	ID of the group of the PBL
moduleid	bigint(10)	ID of the discussion module
instanceid	bigint(10)	Instance number of the discussion module

Table 10.4: **Structure of table mdl_pbl_discuss**

mdl_pbl_forum		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
forumid	int(8)	ID of the forum associated with the PBL module
courseid	int(8)	ID of the course
pblid	int(8)	ID of the PBL
instanceid	bigint(10)	Instance number of the forum module

Table 10.5: **Structure of table mdl_pbl_forum**

mdl_pbl_solution		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
name	mediumtext	Name of the solution
definition	mediumtext	Description of the solution
file_location	text	Location where the associated file will be stored
courseid	bigint(10)	Id of the course
pblid	bigint(10)	Id of the PBL module
userid	bigint(10)	Id of the user who posted the solution
file_name	text	Name of the File

Table 10.6: **Structure of table mdl_pbl_solution**

mdl_pbl_groups		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
name	mediumtext	Name of the group
pblid	bigint(10)	ID of the PBL
courseid	bigint(10)	ID of the course

Table 10.7: **Structure of table mdl_pbl_groups**

mdl_pbl_groups_members		
column name	**Type**	**Purpose**
id	bigint(10)	Auto incremental ID of the table
userid	bigint(10)	ID of the user
groupid	bigint(10)	ID of the group user belongs to

Table 10.8: **Structure of table mdl_pbl_groups_members**

10.3 Development Documentation

Development of this PBL module is also done in the same modular approach (discussed in section 9.3) as *rubric module*. For this module also *access control* is an important issue. So among the different layers ***authorization*** layer is the top most layer. ***Functional logic*** is implemented in between the ***authorization*** and ***database*** layer.

10.3.1 Authorization

For the PBL module we check for the authentication in two different ways,

- Using functions defined in the Moodle libraries
- using the access capability defined in the local *db/access.php* file

Locally defined permissions are used to check if the user is authorized to do something in the current PBL module. To check the permission of a user in any higher context level than the current PBL module, we used the already defined functions in Moodle. Library function which are used in this particular module are,

- ***require_login()***: This function is used to check if the current user is logged in and has the required capability to be in the course.
- ***isstudent()***: This function returns true if the role of the current user is student.
- ***isteacher()***: This function returns true if the role of the current user is teacher.
- ***has_capability()***: This function is used to check if the user have a particular capability in a particular context.

The local capabilities are defined in the *db/access.php* file. This permissions are used in the *has_capability ($capability, $context, $userid)* function to restrict the access of the current user. Here *$context* is the context of the PBL module. *$userid* is only required, if we want to check permission of some user other than the user who is currently logged in. *$capability* names specific to this PBL module are listed below.

- ***view***: It is to authorize users to view the PBL.
- ***createforum***: This will authorize the user to create the forum to discuss the doubts regarding the problem definition.
- ***editRLF***: This is to authorize user to give permission to edit the RLFs.
- ***addRLF***: This is to authorize user to add RLFs.
- ***add_discussion***: To add discussion user need this permission.
- ***view_discussion***: To view discussion user need this permission.
- ***edit_discussion***: This permission will authorize user to edit the discussions.
- ***edit_resources***: This permission will authorize user to update the settings of the file manager.
- ***add_resources***: This is to authorize the user to add, delete, update or share resources with others.
- ***view_solution***: This permission is required to view the solution proposed in the PBL activity.
- ***delete_solution***: This is to authorize the user to delete a solution.

- ***view_rubric***: To view rubrics user need this permission
- ***submit***: To submit the final solution user need this permission.
- ***view_report***: To view the report of the PBL activity user need this permissions.
- ***edit_group***: To edit the group settings user require this permission.
- ***add_rubric***: To add rubric in the PBL activity user require this permission.
- ***add_submission***: User can add final submission link if they have this permission.
- ***view_recent_update***: To view the recent updates user required this permission.

10.3.2 Functional Logic

Now, we can describe the functional logic with respect to the features implemented to support different stages of PBL. Among the different stages, *group formation* and *facilitator allocation* steps are done at the course level, with the help of the features provided by Moodle. After this two steps complete, the next step is *Problem presentation*.

Problem Presentation

To present the problem user need to add a PBL activity module inside a Moodle course. While adding the PBL activity from the Add a activity dropdown menu, user will be redirected to the page *$HOME/course/modedit.php*. In this page we will see a form, which is coded in the local *mod_form.php* file. This form contains four important elements of a PBL activity. Thses are,

- *course*: It is the ID of the current course.
- *name*: It will be stored as the name of the PBL.
- *intro*: It will be stored as the problem definition of the PBL.
- *pbltype*: It is the type of the PBL. It can be any of the four given below (Currently we are not using this categorization in the PBL module, we are using the course level grouping feature),
  - *Individual Project*: In this type of PBL, students individually work on the problem.
  - *Teacher Created Group*: In this type of PBL teacher create the groups of PBL.
  - *Student Created Group*: In this type of PBL students create the groups by their own.

After we fill up the form and save it, records get stored in two different database tables. These are *mdl_course_modules* and the *mdl_pbl*. Once we have added this module we can update or delete it. All the functions to add, update or delete the modules are defined in the local *lib.php* files. These are

- ***pbl_add_instance($pbl)***: Here *$pbl* is an object containing all the data to add a new row in *mdl_pbl* table.
- ***pbl_update_instance($pbl)***: This function updates a PBL instance and *$pbl* is an object as defined above.
- ***pbl_delete_instance($id)***: This functions a PBL instance, having a ID equals to *$id*.

Identification of RLF

For the *RLF*, basically there are three features, *add a RLF*, marking RLF as *known/unknown* and *view RLF*. To add a RLF user needs to fill-up the form, defined in the local *add_rlf.php* file. As user, fill up the form and submit it, records are get stored in the *mod_pbl_pbl_rlf* table. To identify, with which PBL instance a particular RLF belongs, we save the course ID and the PBL ID along with the details in the *mdl_pbl_rlf* table. When the user mark a particular RLF as the known, we insert a row in the *mdl_pbl_rlf_user* table. This table keeps the maping of *userid* with the *rlfid*. When a user mark a already known RLF to unknown, then the entry from the *mdl_pbl_rlf_user* table is deleted.

Now while showing the RLFs in the course homepage, we just check if that RLF belongs to this PBL instance or not. Then to provide the delete button we check if the current user has added the RLF or not. To categorize the RLFs into *known* and *unknown* we check the *mdl_rlf_user* table. If we find an entry in the table, with the current user ID and the RLFID, we show that RLF in the *known* category, otherwise in the *unknown* category.

Discussion

There are two ways we can add discussions in the PBL module. First one is the forum associated with the problem definition and the other one is the different types of discussion teacher can add inside the PBL discussion block. Now all this discussions (*chat*, *forum*, *wiki*) are the activity module of Moodle. We can add all these module inside a course. So the most secure way to add this module inside the PBL module is to use the same functions, which are used to add this module inside the course and keep a track so that we can identify which discussion modules belong to PBL activity. Now all the functions to add this module are called from the *$HOME/course/modedit.php* file. So we created a file with the same name inside the local pbl folder. The code in the file is almost same as the *$HOME/course/modedit.php*. The difference is that we have called the functions to update either the *mdl_discuss*(for the forum associated with problem definition) or *mdl_forum* (other discussions) table inside the local *modedit.php* file. This entry in the pbl database table will associate the modules with the PBL activity. The two functions, we used to update the database tables are,

- ***update_pbl_discuss($courseid,$pblid,$type,$userid,$moduletype)***: This function insert data into the *mdl_discuss* table. Here *$courseid* is the id of the course, *$pblid* is the id of the PBL, *$type* is the name of the module we want to add, *$userid* is the id of the current user and *$moduletype* is the type of the module we want to add.

- ***update_pbl_forum($courseid,$pblid)***: This function is used to create the forum associated with the problem definition. It adds an entry in the *mdl_forum* table. Here *$courseid* is the id of the course, *$pblid* is the id of the PBL.

Now as all these discussion modules are designed to use only inside course, if we add one it will be shown in the course homepage. But we don't want to show it in the course homepage, as we want to use this discussion modules totally as sub-module of the PBL activity.

In a weekly type course, a particular module is added in a particular section. Let the maximum number of the section in the course is x. Now if we insert these modules some section having value $> x$, then modules won't be visible in the course homepage. So in the *$HOME/-mod/pbl/modedit.php* file we find out the maximum number of section for the course, and added the module in a section, having number $> x$.

Resource Sharing

To share the resources we used the *File Manager*[30] block. This block provide the functionalities to have a *My Folder* to store the file for personal use. The other option to upload the files into *group X files* (here *group x* is the name of the group, to which the user belong). So, to use the *File-Manager* plug-in we created the hyper-link to the *MyFolder* and *group x files* using the user-ID and the group id, to which the currently logged in user belongs in the PBL homepage.

Solution Proposal

To propose a solution user needs to fill-up details of the form coded in the local *save_solution.php* file. User can give the solution name, description of the solution and can attach a file with it. To handle the file we have used the Moodle library functions.

- To save the file we used the ***save_files($destination_directory)*** method defined in the *$HOME/lib/formslib.php* file. This method saves the file associated with the form. Here *$destination_directory* is the directory where this file is going to be saved. Now to discriminate among the different solution files, we need to initialize the *$destination_directory* variable accordingly. Here we initialized this variable as ***$CFG→dataroot/$courseid/pbl/--$moduleid/$userid/solution/$solutionid***.
  - *$CFG→dataroot* is address of the Moodle root data directory.
  - *$courseid* is the ID of the course
  - *$moduleid* is the id of the PBL module
  - *$userid* is id of the user who is proposing the solution

  - *$solutionid* is the id of the solution

  Combining all these elements, in this particular order generate a unique position for a particular solution. Details of the solutions are stored in the *mdl_pbl_solution* table.

- Now to view the file associated with the solution we used the ***send_file($path, $filename, $lifetime)*** function. Where *$path* is the directory where the file is, *$filename* is the name of the file and *$lifetime* is the time, the file will be there in cache.

- While deleting a particular solution, the record of that particular solution gets deleted from the *mdl_pbl_solution* table and to delete the associated file we used the function ***fulldelete($filelocation)***, where *$filelocation* is the location of the file we want to delete.

Both the *fulldelete()* and *send_file()* are defined in the *$HOME/lib/filelib.php* file.

Solution Submission

For the solution submission we have used the *assignment module*[31] of Moodle. We used the same procedure of adding this module as sub-module of PBL as addition of the discussion modules, which we already have described in subsection *Discussion*. Creation of new submission will add a new row to the *mdl_discuss* table.

Evaluation

For evaluation, we can add different types of *rubric*. Now as the rubric is also an activity module in Moodle, we handled the addition of rubrics in the same way as the addition of different discussion module.

Now to support in the *evaluation* there is a feature to generate *Report*. Now to generate report about the PBL activity, we need to process data from many tables. To get the overall report of a group, we first find out the members of the group, then we accumulate data for all the user. The tables which are accessed to generate the report are,

- *groups_members*: To access the information about the members of the groups.

- *pbl_rlf*: To get the number of total RLFs proposed by the members of the group.

- *pbl_solution*: To get the number of the solutions proposed in the PBL by the group.

- *pbl_discuss*: To get the numbers of discussions added in the PBL by the group.

- *forum_discussions*: To get the number of discussion topics in the forums in the PBL by the group.

- *forum_posts*: To get the number of posts in the forum.

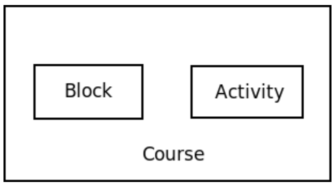


Figure 10.17: Hierarchy of course, activity and blocks in Moodle

- *chat_messages*: To get the number of chat messages passed in the PBL by the group.

To get the individual records we use the same tables, to extract the numbers using the user's id for whom we want to see the report.

10.3.3 Database layer

For the database layer in the PBL module we used all the functions we have used to implement rubrics module's database layer. In addition to the functions described in the section 9.3.3, we have used the

- ***record_exists($table, $field1, $value1, $field2, $value2, $field3, $value3)***: This function returns true, if some record exists in the table named *$table*, where *$filed1* is having the value *$value1* and so on. For this function *$field1*, *$value1*, *$field2*, *$value2*, *$field3*, *$value3* are the optional arguments.

All these functions are defined in the *$HOME/lib/dmllib.php* file.

10.4 Challenges

10.4.1 Moodle Structure

The basic installation of Moodle 1.9.x will create almost 6500 files in the local directory. The library files which are used frequently, like *moodlelib.php*, *weblib.php*, *accesslib.php* each having more than 5000 lines of code. Therefore, Moodle is large software, and initially it takes time to figure out how a particular feature is working internally. Other than this, before developing a module we need to know *basic databases structure of Moodle*, *how access is controlled in Moodle*, *how the XMLDB editor works*, *about the directory structure of a activity module*, *how the Moodle quick-form extension works* etc.

10.4.2 Include others activity module inside the PBL module

In Moodle *course*, *activity module* and *blocks* have a hierarchy (Figure 10.17). So we can easily add a block or an activity module inside a course, using the APIs already defined in the library files of Moodle. But we cannot add a block inside another block, same applies for activity module. But for the PBL module we need to use some of the blocks and activity modules of Moodle. For example the

- Forum
- Chat
- Wiki
- Participants
- Assignment

modules/blocks are necessary for the PBL module. Now, we can achieve this by updating the database tables, which gets updated while we add a particular module inside a course. Generally two database tables of Moodle gets updated while we add a module inside a course. These are ***mdl_course_modules*** and ***mdl_$module***, *$module* is the name of the module we want to add.

But this is too tricky to modify these two tables directly. Because mainly of two reasons. First, we may add invalid data into the tables as we are not using proper API, and the second is structure of the tables of different activity module is different. So we need to write a function for each module. So this solution is not a practical one.

The second solution is to use the same APIs, which are used to add modules inside a course. Now to add a module inside a course the steps are stated below.

- Choose the particular activity module in the drop-down box *Add an activity...* inside the course homepage.
- From course homepage user will be redirected to *course/modedit.php*
- User have to fill up the form shown in the *course/modedit.php* file and submit

So, if we can update the *mdl_pbl_discuss* table, along with the other tables which gets updated when we submit the form in the *course/modedit.php*, then we would be able to track which activity modules are associated with the PBL module. Hence, we have to add code inside the *course/modedit.php* file. But adding code directly in the *$HOME/course/modedit.php* is not a good idea. Because, we are modifying core Moodle files and this will always update the *mdl_pbl_discuss* table, even if we add the new activity module for the course not for the PBL module. So we declare a *modedit.php* file inside the PBL module and inserted our code to update the *mdl_pbl_discuss*. Now whenever user adds a particular module inside the PBL module, user will be redirected to the *$HOME/mod/PBL/modedit.php* file.

The method stated above is successful one. We could have achieved the same, by just updating the table *mdl_PBL_discuss*, while we try to add a module inside PBL and then redirect to *course/modedit.php*. But if then the user cancel the module addition, there is no way to roll back the update happen in the *mdl_PBL_discuss* table.

10.4.3 File Up-loader

Moodle 1.9.x and the previous versions do not have any feature, such that the students can upload files whenever they want and maintain a **MyFolder**. But file upload facility for student is a well desired feature for the PBL module. So either we had to write a block which will have all the desired features to manage files else we need to use some plug-in of Moodle. In Moodle there are few file management plug-in available, like *File manager*, *Repository File Manager*, *Users Upload Hack* etc. So we needed to test all these plug-ins and find out which one suits best for the PBL module. So among these plug-ins we found-out that, the *File Manager* blocks suits most for PBL, as this has the feature to manage files in group level.

10.5 Future Work

- Currently there are only two types of role in the PBL module, *Teacher* and *Student*. But there should be a third role as proposed in the chapter 6, i.e. *Facilitator*. Permission of facilitator, will be in between the permission of teacher and student. A user of role teacher exclusively will be able to create PBL activities and assign facilitator in a particular PBL activity for a group. A facilitator will have the similar permissions like the teacher in a PBL activity, where he is assigned as a facilitator.

- Currently grouping is done at the course level and there is no option such that students can create the group of their own choice. Hence, grouping should be done at the PBL activity level and option should be provided so that students will be able to create the groups. Currently it is not mandatory to create the groups before problem presentation. But creating group must be made mandatory before the teacher can present the problem.

- For communication and discussion currently user can use *chat*,*forum* or *wiki*. But for better communication *audio/video chat*, *conferencing*, *whiteboard* features should be added.

- The sub-activity and sub-modules in the PBL modules are not well structured. It is important to give them some similar structure like the block or activity in Moodle, so that developers can easily extend the features and add new functionalities.

- As described in the chapter 6, *recent activity* module feature need to be developed. This will help teacher to keep track with the progress of the group.

- In the current PBL module file sharing feature is implemented in the course level, so user can gain access to the files from anywhere in the course. This file sharing feature need to be implemented at the PBL activity level.

- Currently we have implemented many functionality directly in file where we need that, where we could have written functions in the *lib.php* file to achieve the same action. This will make the code more modular.

- In the current version of PBL module if the user tries to delete any sub-module, like *RLF*, *solution* no warning is shown, but it is a desired functionality.
- Rating feature for the solutions proposed, forum posts, RLF documented and file shared should be there.

Chapter 11

Conclusion

We have developed two activity modules as the part of this project. One is the PBL module and the other is the rubrics module. The *PBL* module is a complete tool to support PBL, as it has feature using which users of PBL can carry-out all the steps described in section 2. So it can be used in distance learning scenario as well as in blended-learning environment. The *rubrics* module has been developed for easy creation and processing of well known assessment tool rubric. Now, testing the usability of the PBL module is the main task remains. For usability test some controlled experiment need to be done. Controlled experiment can be designed in the following manner. In a course there will be two PBL activities, one will be carried out using only Moodle and other one will be carried out using this new PBL module in Moodle. Then by the comparing the student's performance, effort needed from instructor to carry-out PBL activity in both the cases etc. Questioner can also be created to get the feedback from both the faculty and students.

Building these tools as plug-in for the learning management system Moodle had some advantages, like we can use already developed features (forum, wiki) in Moodle. But initially it took plenty of time, understanding how Moodle works and if we want to add a functionality how to do it. For the PBL module, we divided the *problem based learning process* into different steps and for each step we developed some functionalities. This way the development has been done in a modular approach.

Bibliography

[1] http://docs.moodle.org/20/en/File.

[2] http://moodle.org/mod/data/view.php?d=13&rid=107&filter=1.

[3] http://www.kem.edu/dept/METC/PBL%20Ramnarayan.pdf.

[4] http://moodle.org/.

[5] http://moodle.org/mod/forum/discuss.php?d=43776.

[6] The Higher Education Statistics Agency. http://www.hesa.ac.uk/.

[7] A. Al-Ajlan and H. Zedan. Why Moodle. In *Future Trends of Distributed Computing Systems, 2008. FTDCS'08. 12th IEEE International Workshop on*, pages 58–64. IEEE, 2008.

[8] I.E. Allen and J. Seaman. Making the grade: Online education in the United States, 2006. *Needham, MA: Sloan Consortium*, 2006.

[9] Means B, Toyama Y, Murphy R, Bakia M, and Jones K. Evaluation of evidence-based practices in online learning: A meta-analysis and review of online learning studies. Aug, 2009.

[10] T. Barrett. Understanding Problem-Based Learning.

[11] D.R. Brodeur, P.W. Young, and K.B. Blair. Problem-based learning in aerospace engineering education. In *Proceedings of the 2002 American Society for Engineering Education Annual Conference and Exposition, Montreal, Canada*, pages 16–19. Citeseer, 2002.

[12] J. Chen, M. Lee, H. Lee, Y. Wang, L. Lin, and J. Yang. An online evaluation of problem-based learning (PBL) in Chung Shan Medical University, Taiwan-a pilot study. *ANNALS-ACADEMY OF MEDICINE SINGAPORE*, 35(9):624, 2006.

[13] A. Ellis, L. Carswell, A. Bernat, D. Deveaux, P. Frison, V. Meisalo, J. Meyer, U. Nulden, J. Rugelj, and J. Tarhio. Resources, tools, and techniques for problem based learning in

computing. In *Working Group reports of the 3rd annual SIGCSE/SIGCUE ITiCSE conference on Integrating technology into computer science education*, pages 41–56. ACM, 1998.

[14] Rosenthal H Gallagher S.A., Stepien W.J. The effects of problem-based learning on problem solving. *Gifted Child Quarterly*, 36(4):195–200, 1992.

[15] R. Garcia-Robles, S. Vicente-Diaz, and A. Linares-Barranco. An eLearning Standard Approach for Supporting PBL in Computer Engineering. *Education, IEEE Transactions on*, 52(3):328–339, 2009.

[16] H.G. SCHMIDT J.H.C. MOUST, H.J.M. VAN BERKEL. Signs of erosion: Reflections on three decades of problem-based learning at maastricht university. *Higher Education*, 50(4):665–683, Oct, 2005.

[17] G.E. Lautenbacher, J.D. Campbell, B.B. Sorrows, and D.E. Mahling. Supporting collaborative, problem-based learning through information system technology. In *Frontiers in Education Conference, 1997. 27th Annual Conference.'Teaching and Learning in an Era of Change'. Proceedings.*, volume 3, pages 1252–1256. IEEE, 2002.

[18] N. Linge and D. Parsons. Problem-based learning as an effective tool for teaching computer network design. *Education, IEEE Transactions on*, 49(1):5–10, 2006.

[19] J. Macías-Guarasa, R. San-Segundo, J.M. Montero, J. Ferreiros, and R. Córdoba. Tools and strategies for improving PBL laboratory courses with a high student-to-Faculty ratio. In *Frontiers in Education, 2005. FIE'05. Proceedings 35th Annual Conference*, pages F2C–7. IEEE, 2006.

[20] M. Qiu and L. Chen. A Problem-Based Learning Approach to Teaching an Advanced Software Engineering Course. In *2010 Second International Workshop on Education Technology and Computer Science*, pages 252–255. IEEE, 2010.

[21] I. Richardson and Y. Delaney. Problem Based Learning in the Software Engineering Classroom. In *Software Engineering Education and Training, 2009. CSEET'09. 22nd Conference on*, pages 174–181. IEEE, 2009.

[22] C.K. Riesbeck. Designing Web-Based Interactive Learning Environments for Problem-Based Learning. In *Proceedings of the Fifth IEEE International Conference on Advanced Learning Technologies*, pages 333–337. IEEE Computer Society, 2005.

[23] Gu Yue-Sheng Wang Jian-Ping, Gao Guo-Hong. Building up problem-based learning platform based on j2ee. *Education Technology and Computer Science, 2009. ETCS '09. First International Workshop on*, 2:614–616, Mar, 2009.

[24] R. Waters and M. McCracken. Assessment and evaluation in problem-based learning. In *Frontiers in Education Conference, 1997. 27th Annual Conference.'Teaching and Learning in an Era of Change'. Proceedings.*, volume 2, pages 689–693. IEEE, 2002.

[25] J. Zumbach, D. Kumpf, and S.C. Koch. Using multimedia to enhance problem-based learning in elementary school. *Information technology in childhood education annual*, 25:37, 2004.

[26] http://docs.moodle.org/20/en/Context.

[27] http://docs.moodle.org/20/en/Wiki_module.

[28] http://docs.moodle.org/20/en/Chat_module.

[29] http://docs.moodle.org/20/en/Forum.

[30] http://docs.moodle.org/20/en/File_manager_block.

[31] http://docs.moodle.org/20/en/Assignment_module.

[32] http://docs.moodle.org/dev/Roles#Context.

[33] http://www.csub.edu/TLC/options/resources/handouts/Rubric_Packet_Jan06.pdf.